# Technical Notes On:

# A5Storage

Last Updated: 10/30/2014 11:17:00 AM

# Contents

# Overview

A5Storage is both the Alpha Five root namespace and the assembly name for an object that accesses data from a container without the caller needing to explicitly know where that container is or how the data is accessed. Client code passes a connection string to the implementation; which uses the information in that string to perform the requested operations.

There are currently three implementations of A5Storage:

| | |
|---|---|
| ***Disk*** | Uses disk functions to read and write a file based object |
| ***Azure*** | Uses the .NET API to access Azure Blob Storage objects |
| ***AmazonS3*** | Uses the Amazon AWS client .NET assembly to access S3 objects |

For performance reasons, many of the functions to reference a storage object will not attempt to connect to the server until a function is called (and only if that function needs to return data from the remote server). Because connections must be reestablished for each call, errors may occur at any time. Do not assume that a successful return from an Open or Reference function means that the container exists or that a connection has been successful.

There are two objects you will interact with. The **Container** can get and set object data and may be all you need. The **DataItem** is used when you want to create a reference to an object (without necessarily reading the data) and get information about it.

# Connection Strings

To keep the user of A5Storage agnostic about the storage type, the storage container is opened using a connection string similar to a SQL database connection string. Changing the connection string changes the storage used. A connection string dialog is included in the assembly in order to make it easier to create connection strings at development time or to prompt for information when configuring a server.

## *Connection String Properties*

Connection strings will differ depending on the implementation used. The syntax of the connection string is simply a list of property value assignments of the form:

> <property name>='<value.';…
>
> **For example**:
>
> "Provider='Disk';Container='C:\A5WebRoot';"
>
> Note: The single quotes around the value and the terminating semi-colon for each item are required.

One common property is the provider itself. The keyword for the provider is, of course, "Provider". Its value may be 'Azure', 'AmazonS3' or 'Disk'.

A second property available is Timeout. This property is the number of milli-seconds a request should be limited to. Note: Azure and AmazonS3 are the only providers that currently support the timeout parameter explicitly, but it can be set for all providers.

### Disk

Disk storage may be either local to the machine or available over the network (using UNC names).

In addition, you may specify a local or domain user account and password to be used to access the path you specify.

#### Examples

"Provider='Disk';Container='C:\A5WebRoot';"
"Provider='Disk';Container='\\MyServer\MyShare\A5WebRoot';Account='Bob';Password='Secret';"

The properties available for the Disk provider are:

- Provider
- Container
- Account (optional)
- Password (optional)
- Timeout (optional)

## Azure

For Azure storage you will need to have your account and access key as well as the container name.  The connection string format is:

### Example

"Provider='Azure';Account='MyAccount';AccessKey='MyKey';Container='MyContainer';"

The properties available for the Azure provider are:

- Provider
- Account
- AccessKey
- Container
- Timeout (optional)

## Azure Test Storage

For Azure storage, there is an additional option, the test container.  If you have installed the Azure SDK and want to test with Azure test storage, you can do so either through the dialog or by setting the connection string as follows:

### Example

"Provider='Azure';UseDevelopmentStorage='true';Container='MyContainer>';"

The properties available for the Azure provider when using test storage are:

- Provider
- UseDevelopmentStorage
- Container
- Timeout (optional)

## AmazonS3

For Amazon S3 storage you will need to have your access key and your secret key as well as the container name.  The connection string format is:

### Example

"Provider='AmazonS3';AccessKey='MyKey';SecretKey='MySecret';Container='MyContainer';"

The properties available for the Azure provider are:

- Provider
- Account
- AccessKey
- Container
- Timeout (optional)

## How it All Works

In order to access remote data, you first create a connection string (as discussed above). Next you create a reference to a container by calling the static container method Open(). The container reference can be used to access items inside of it. If you want to hold on to a reference to an item, the container provides a method to do so.

For example, the script below opens a container (a disk folder in this case) and retrieves an object and its content type:

```
dim ConnectionString as C = "Provider='Disk';Container='C:\A5WebRoot';"
dim Container        as A5Storage::DataContainer = null_value()
dim Item             as A5Storage::DataItem = null_value()

CallResult = A5Storage::DataContainer::Open(Container,  \
                    "Provider='Disk';Container='c:\A5Webroot';")
?CallResult.Success

? Container.ReferenceItem(Item, "speak.html")
?Container.CallResult.Success
.T.

?Item.ModifiedTime
= 09/29/2010 09:44:59 41 pm
?Item.ContentType
= "text/html"
?Item.Size
= 362

dim Text AS C
dim ContentType as C
Item.Get(Text, ContentType)
?Text
= <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<HTML xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<HEAD>
<BODY>
Speaking a message.
<EMBED src="Message.wav" autostart=true loop=false volume=100
hidden=true><NOEMBED><BGSOUND src="Message.wav"></NOEMBED>
</BODY>
</HTML>

?ContentType
= "text/html"
```

# Error Handling

## *Important – A5Storage Objects are State-full!*

Do not share A5Storage objects across threads! Share the connection string for the container and the item name, but create a new object every time. Most functions save the error state as a member variable and return a Boolean true or false to indicate success or failure. If you share and object and two different users make overlapping calls, the status of one of the calls may be overwritten.

In order to work better with XBasic, the C# classes return either a Boolean (instance functions) or a CallResult (static functions) with detailed error information.

Again, **DO NOT SHARE INSTANCES OF A5Storage::DataContainer OR A5Storage::DataItem ACROSS THREADS!** The instance functions return a Boolean true or false and hold the result of the last error in the CallResult property. If you make a second function call, the result of the last call is lost.

The "right" way to call functions from XBasic:

Instance:

```
dim Result as CallResult
if .not. Item.Myfunction(…..) ' check for failure
        Result = Item.CallResult ' save the call result off
```

Static:

```
Result = Container.Function(…)
```

# The A5Storage Namespace

## *A5Storage::DataContainer*

This class manages access to a specific data container. This can be an Azure Blob Storage account, an Amazon S3 Storage bucket or a folder on disk. It acts as the reference point from which other objects are referenced.

Notes:
1. Object names can include paths (indicated with forward slashes '/').
2. If you are referencing an object on disk, you can include a path that will act as the root of all other objects. The object path will be the concatenation of the disk path of the container and the path embedded in the storage name (as a relative path).

### Properties

| Property | Type | Example Content |
|---|---|---|
| CallResult | P | |
| ConnectionString | C | " Provider='Disk';Container='C:\A5WebRoot';" |
| Exists | C | Returns true if the container exists. |
| Name | C | "C:\A5WebRoot" |
| StorageType | C | "Disk" |

### CopyContentsTo

This function will copy all items to another container. Any existing items will be replaced.

CopyNewContentsTo AS L (TargetContainer as A5Storage::DataContainer)

**Example**

```
dim Container              as A5Storage::DataContainer = null_value()
dim TargetContainer     as A5Storage::DataContainer = null_value()

CallResult = A5Storage::DataContainer::Open(SubContainer,  \
                          "Provider='Disk';Container='c:\A5Webroot\logs';")
?CallResult.Success
CallResult = A5Storage::DataContainer::Open(TargetSubContainer,  \
                          "Provider='Disk';Container='c:\A5Webroot2\logs';")
?CallResult.Success
?Container.CopyContentsTo(TargetContainer)
```

## CopyNewContentsTo

This function will copy all items that are newer to another container.

CopyNewContentsTo AS L (TargetContainer as A5Storage::DataContainer)

**Example**

```
dim Container              as A5Storage::DataContainer = null_value()
dim TargetContainer     as A5Storage::DataContainer = null_value()

CallResult = A5Storage::DataContainer::Open(Container,  \
                          "Provider='Disk';Container='c:\A5Webroot\logs';")
?CallResult.Success
CallResult = A5Storage::DataContainer::Open(TargetContainer,  \
                          "Provider='Disk';Container='c:\A5Webroot2\logs';")
?CallResult.Success
?Container.CopyNewContentsTo(TargetContainer)
```

Note:  This function is intended for pushing information from local (disk) storage to cloud storage. Its first application is to push log files from a local machine directory to the cloud.

## DecryptConnectionString

Use this function to decrypt a connection string if you have the passphrase (or the passphrase is defaulted).

```
STATIC DecryptConnectionString AS CallResult (
                              BYREF DecryptedString    AS C,
                              ConnectionStringIn       AS C
                              [, PassphraseIn          AS C])
```

**Example**

```
dim cr as CallResult
dim ConnectionString    as C
dim ResultString        as C
cr = A5Storage::DataContainer::DecryptConnectionString(   ResultString,
                                                          ConnectionString)
?cr.Success
```

## CopyItemTo

This function will copy an item to another container. The function is optimized to use native methods to copy between the same types of containers when possible.

CopyItemto AS L ( TargetContainer as A5Storage::DataContainer,
ItemName AS C)

**Example**

```
dim ConnectionString as C = "Provider='Disk';Container='C:\A5WebRoot';"
dim Container                as A5Storage::DataContainer = null_value()
dim TargetContainer          as A5Storage::DataContainer = null_value()

CallResult = A5Storage::DataContainer::Open(Container,  \
                             "Provider='Disk';Container='c:\A5Webroot';")
CallResult = A5Storage::DataContainer::Open(TargetContainer,  \
                             "Provider='Disk';Container='c:\A5Webroot2';")
?CallResult.Success
?Container.CopyItemTo(TargetContainer, "MyTargetPath/MyObject")
```

## DeleteItem

This function will delete an object. If the object does not exist, no error will be returned.

DeleteItem AS L (TargetPath AS C)

**Example**

```
dim ConnectionString as C = "Provider='Disk';Container='C:\A5WebRoot';"
dim Container      as A5Storage::DataContainer = null_value()
dim Item                   as A5Storage::DataItem = null_value()

CallResult = A5Storage::DataContainer::Open(Container,  \
                             "Provider='Disk';Container='c:\A5Webroot';")
?CallResult.Success
?Container.DeleteItem("MyTargetPath/MyObject")
```

## EncryptConnectionString

Use this function to encrypt a connection string.

Note: If you do not provide a passphrase then it will be defaulted. Only this assembly will be able to decrypt the string, but anyone with the assembly will be able to decrypt it.

```
STATIC EncryptConnectionString AS CallResult (
                                    BYREF EncryptedString    AS C
                                    ConnectionStringIn       AS C
                                    [, PassphraseIn          AS C])
```

**Example**

```
dim cr as CallResult
dim ConnectionString    as C = "Provider='Disk';Container='C:\A5WebRoot';"
dim ResultString        as C
cr = A5Storage::DataContainer::EncryptConnectionString(   ResultString,
                                                          ConnectionString,
                                                          "secret");
```

## GetItem

Retrieves an item from the container using the name you provide. The content type is also returned.

Note: You must dimension the resulting Data variable and the ContentType variable prior to making the call.

```
GetItem AS L (      BYREF Data          AS B,
                    BYREF ContentType   AS C,
                    TargetPath          AS C)

GetItem AS L (      BYREF Data          AS C,
                    BYREF ContentType   AS C,
                    TargetPath          AS C)
```

**Example**

```
dim ConnectionString    as C = "Provider='Disk';Container='C:\A5WebRoot';"
dim ResultData          as B
dim ContentType         as C
dim Container           as A5Storage::DataContainer = null_value()

CallResult = A5Storage::DataContainer::Open(Container,  \
                        "Provider='Disk';Container='c:\A5Webroot';")
?CallResult.Success

?Container.GetItem(ResultData, ContentType, "MyFile.jpg")
```

## ListItems

Lists the contents of the container as a CRLF() delimited string.

```
ListItems AS L (BYREF String as C [, SearchPrefix as C])
```

**Example**

```
dim ConnectionString    as C = "Provider='Disk';Container='C:\A5WebRoot';"
dim ResultString        as C
dim Container           as A5Storage::DataContainer = null_value()
dim Item                as A5Storage::DataItem = null_value()

CallResult = A5Storage::DataContainer::Open(Container,  \
                        "Provider='Disk';Container='c:\A5Webroot';")
?CallResult.Success
?Container.ListItems(ResultString)
Showvar(ResultString)
```

## Open

Opens (references) a storage container based on a connection string.

Note: If the connection string is encrypted, the passphrase must be provided or the container will attempt to decrypt the connection string using the default passphrase. See the function EncryptConnectionString for more information.

```
STATIC Open AS CallResult (
        BYREF       NewContainer        AS A5Storage::DataContainer,
                    ConnectionStringIn  AS C
                    [, Passphrase       AS C])

STATIC Open AS CallResult (
        BYREF       NewContainer        AS A5Storage::DataContainer,
                    Cache               AS A5Storage::NameCache,
                    ConnectionStringIn  AS C
                    [, Passphrase       AS C])
```

**Examples**

```
dim CallResult   as CallResult
dim Container    as A5Storage::DataContainer = null_value()

CallResult = A5Storage::DataContainer::Open(Container,  \
                                "Provider='Disk';Container='c:\A5Webroot';")
if CallResult.Success
        …
```

**OR**

```
dim CallResult   as CallResult
dim Container    as A5Storage::DataContainer = null_value()
dim Storages as C = <<%txt%
        Storage1        Provider='Disk';Container='C:\A5WebRoot1';
        Storage2        Provider='Disk';Container='C:\A5WebRoot2';
        %txt%
dim Cache as A5Storage::NameCache = new A5Storage::NameCache(Storages)


CallResult = A5Storage::DataContainer::Open(Container,  Cache,\
                                "::NAME::MyConnectionString")
if CallResult.Success
        …
```

Note: The connection string passed in the second example does NOT have to be a named connection. If you DO pass a named connection, you must also provide a cache to look it up in.

## ReferenceItem

Creates and returns a reference to an item within the container.

Notes:
1. You must dimension Item before making the call.
2. No data is actually read until you ask for it.

```
ReferenceItem AS L (BYREF    Item          AS A5Storage::DataItem,
                             TargetPath    AS C)
```

**Example**

```
dim CallResult   as CallResult
dim Container    as A5Storage::DataContainer = null_value()

CallResult = A5Storage::DataContainer::Open(Container,  \
                                        "Provider='Disk';Container='c:\A5Webroot';")
if CallResult.Success
        Dim Item A5Storage::DataItem = null_value()
        Container.ReferenceItem(Item, "MyObjectName")
        …
```

## SaveItemToFile

SaveItemToFile AS L (SourcePath AS C, TargetPath AS C)

**Example**

```
dim CallResult   as CallResult
dim Container    as A5Storage::DataContainer = null_value()

CallResult = A5Storage::DataContainer::Open(Container,  \
                                        "Provider='Disk';Container='c:\A5Webroot';")
if CallResult.Success
        ?Container.SaveItemToFile("MyItempath/MyItem.jpg", \
                                        "c:\temp\MyItem.jpg")
        …
```

## SetItem

SetItem AS L (Source AS B, ContentType AS C, TargetPath AS C)
SetItem AS L (Source AS C, ContentType AS C, TargetPath AS C)
SetItem AS L (Source AS System::IO::Stream,
ContentType AS C, TargetPath AS C)

**Example**

```
dim CallResult   as CallResult
dim Container    as A5Storage::DataContainer = null_value()

CallResult = A5Storage::DataContainer::Open(Container,  \
                                        "Provider='Disk';Container='c:\A5Webroot';")
if CallResult.Success
        ?Container.SetItem("<html><body>Hello World</body> </html>",  \
                                "text/html",  \
                                "MyPath/ MyItem.html")
        …
```

## SetItemFromFile

SetItemFromFile AS L ( SourcePath AS C,
ContentType AS C, TargetPath AS C
[, Offset as N = 0 [, Length as N = -1]])

**Example**

```
dim CallResult   as CallResult
dim Container    as A5Storage::DataContainer = null_value()

CallResult = A5Storage::DataContainer::Open(Container,  \
                                        "Provider='Disk';Container='c:\A5Webroot';")
if CallResult.Success
        ?Container.SetItemFromFile ( "c:\temp\MyItem.jpg", \
                                        "MyItempath/MyItem.jpg")
        …
```

## WriteItemToStream

Retrieves an item from the container using the name you provide. The content type is also returned. The item is written to the stream you provide.

Note: You must dimension the resulting Stream, Length and ContentType variables prior to making the call.

```
WriteItemToStreamStream AS L (
                                Stream      AS System::IO::Stream
                        BYREF   Length      AS N,
                        BYREF   ContentType AS C,
                                TargetPath  AS C)
```

**Example**

```
dim Stream              as System::IO::Stream = null_value()
dim ContentType         as C
dim Length              as N
dim Buffer              as B
dim CallResult  as CallResult
dim Container   as A5Storage::DataContainer = null_value()

Stream = new System::IO::MemoryStream()
CallResult = A5Storage::DataContainer::Open(Container,  \
                                "Provider='Disk';Container='c:\A5Webroot';")
if CallResult.Success
        if Container.WriteItemToStream(Stream, Length, ContentType, "Speak.a5w")
                Stream.Seek(0, System::IO::SeekOrigin::Begin)

                dim Reader as System::IO::BinaryReader = \
                        new System::IO::BinaryReader(Stream)
                Buffer = Reader.ReadBytes(Length)
                Stream.close()
        else
                CallResult = Container.CallResult
        end if
end if

if CallResult.Success
        showvar("Type: " + ContentType + crlf() + "Length: " + Length + crlf(2) + buffer,  \
                "Success")
else
        showvar(CallResult.Text, "Error")
end if
```

## ShowConnectionStringDialog

This function displays a connection string dialog and returns a clear text or encrypted password. The function returns a call result which indicates whether the dialog was exited by clicking OK or Cancel, or if an error occurred in decrypting/encrypting the connection string.

The EncryptResponse parameter is used to set the initial value of the Encrypt Connection String checkbox.

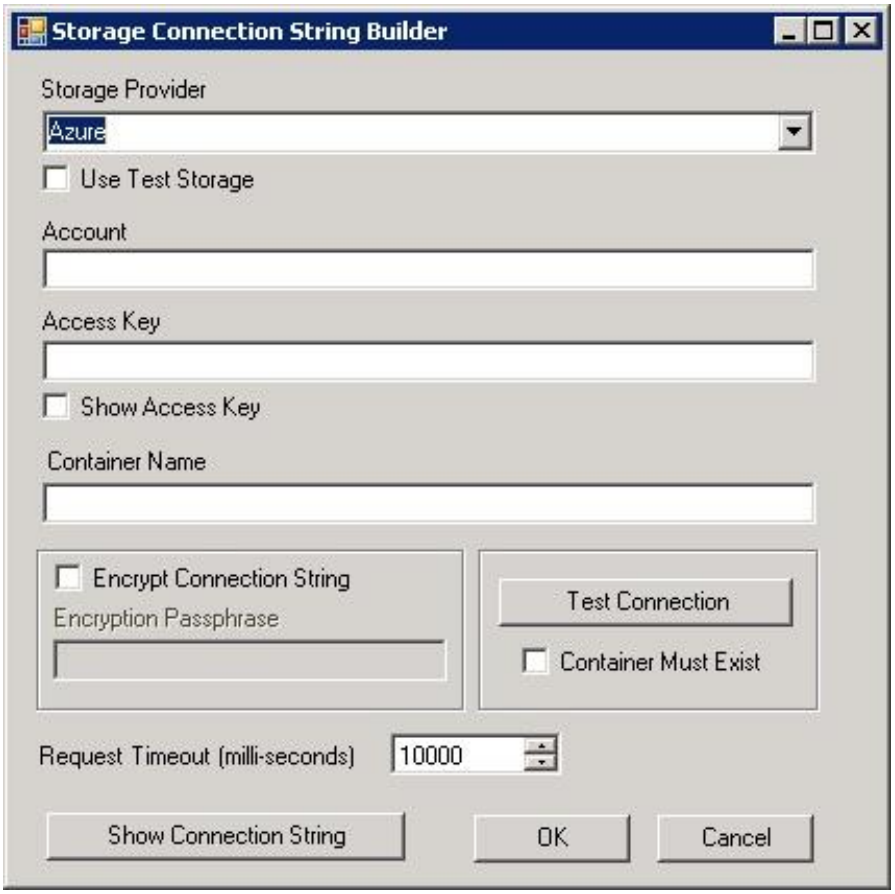
The Passphrase parameter is used to decrypt the connection string (if it is encrypted). If the decryption succeeds, the value is used to set the initial value of the Encryption Passphrase text box. If the decryption fails (or a passphrase is not provided), a dialog prompts the user for a passphrase until they either enter the correct one, or they click the Cancel button.

The user can check or uncheck the Encrypt Connection String checkbox or the text value of Encryption Passphrase before returning from the dialog. If the checkbox is set and OK is clicked, the connection string is encrypted using the current value of the Encryption Passphrase text box.

```
STATIC ShowConnectionStringDialog AS CallResult (
                    BYREF    ConnectionStringResult    AS C
                             ConnectionStringIn        AS C
                             [, EncryptResponse        AS L
                             [, Passphrase             AS C]])
```

**Example**

```
dim ConnectionString as C
dim CR as CallResult
CR = A5Storage::DataContainer::ShowConnectionStringDialog(
ConnectionString,
                                                    ConnectionString,
                                                    .t., "mysecret")
```


Storage Connection String Builder
Storage Provider
Azure
Use Test Storage
Account
Access Key
Show Access Key
Container Name
Encrypt Connection String
Encryption Passphrase
Test Connection
Container Must Exist
Request Timeout (milli-seconds)
10000
Show Connection String
OK
Cancel

## *A5Storage::SubContainer*

This class manages access to a specific "folder" or "sub-path" within a data container. As with the DataContainer object, it acts as the reference point from which other objects are referenced below it in the hierarchy. It does that by maintaining a "prefix" or "path" if you will, which all references are assumed to include.

Notes:
1. If you are referencing an object on disk, the prefix is intended to be a folder or partial folder path. The full object path will be the concatenation of the disk path of the container, the prefix and the path embedded in the storage name (as a relative path).
2. A slash ("/") is added to each part of the container/sub-container hierarchy to form the full path to the storage object.

### Properties

| Property | Type | Example Content |
|---|---|---|
| CallResult | P | |
| Prefix | C | "logs" – a partial path |
| Container | P | Parent Container |

### CopyContentsTo

This function will copy all items to another subcontainer. Any existing items will be replaced.

CopyNewContentsTo AS L (TargetContainer as A5Storage::SubContainer)

**Example**

```
dim SubContainer                as A5Storage::SubContainer = null_value()
dim TargetSubContainer          as A5Storage::SubContainer = null_value()

CallResult = A5Storage::SubContainer::Open(SubContainer,  \
                        "Provider='Disk';Container='c:\A5Webroot';", \
                        "logs")
?CallResult.Success
CallResult = A5Storage::SubContainer::Open(TargetSubContainer,  \
                        "Provider='Disk';Container='c:\A5Webroot2';", \
                        "logs")
?CallResult.Success
?Sub
```
**Container.CopyContentsTo**(TargetSubContainer)

## CopyNewContentsTo

This function will copy all items that are newer to another subcontainer.

CopyNewContentsTo AS L (TargetContainer as A5Storage::SubContainer)

**Example**

```
dim SubContainer                as A5Storage::SubContainer = null_value()
dim TargetSubContainer          as A5Storage::SubContainer = null_value()

CallResult = A5Storage::SubContainer::Open(SubContainer,  \
                        "Provider='Disk';Container='c:\A5Webroot';", \
                        "logs")
?CallResult.Success
CallResult = A5Storage::SubContainer::Open(TargetSubContainer,  \
                        "Provider='Disk';Container='c:\A5Webroot2';", \
                        "logs")
?CallResult.Success
?Sub
```
**Container.CopyNewContentsTo**(TargetSubContainer)

Note: This function is intended for pushing information from local (disk) storage to cloud storage. Its first application is to push log files from a local machine directory to the cloud.

## CopyItemTo

This function will copy an item to another subcontainer. The function is optimized to use native methods to copy between the same types of containers when possible.

CopyItemTo AS L ( TargetContainer as A5Storage::SubContainer,
ItemName AS C)

**Example**

```
dim SubContainer                as A5Storage::SubContainer = null_value()
dim TargetSubContainer          as A5Storage::SubContainer = null_value()

CallResult = A5Storage::SubContainer::Open(SubContainer,  \
                        "Provider='Disk';Container='c:\A5Webroot';", \
                        "logs")
CallResult = A5Storage::SubContainer::Open(TargetSubContainer,  \
                        "Provider='Disk';Container='c:\A5Webroot2';", \
                        "logs")
?CallResult.Success
?Sub
```
**Container.CopyItemTo**(TargetSubContainer, "MyTargetPath/MyObject")

## DeleteItem

This function will delete an object. If the object does not exist, no error will be returned.

DeleteItem AS L (TargetPath AS C)

**Example**

```
dim ConnectionString    as C = "Provider='Disk';Container='C:\A5WebRoot';"
dim SubContainer        as A5Storage::SubContainer = null_value()
dim Item                as A5Storage::DataItem = null_value()

CallResult = A5Storage::SubContainer::Open(SubContainer,  \
                        "Provider='Disk';Container='c:\A5Webroot';", \
                        "logs")
?CallResult.Success
?SubContainer.DeleteItem("MyTargetPath/MyObject")
```

## GetItem

Retrieves an item from the container using the name you provide. The content type is also returned.

Note: You must dimension the resulting Data variable and the ContentType variable prior to making the call.

```
GetItem AS L (      BYREF Data          AS B,
                    BYREF ContentType   AS C,
                    TargetPath          AS C)

GetItem AS L (      BYREF Data          AS C,
                    BYREF ContentType   AS C,
                    TargetPath          AS C)
```

**Example**

```
dim ConnectionString    as C = "Provider='Disk';Container='C:\A5WebRoot';"
dim ResultData          as B
dim ContentType         as C
dim SubContainer        as A5Storage::SubContainer = null_value()

CallResult = A5Storage::SubContainer::Open(Container,  \
                        "Provider='Disk';Container='c:\A5Webroot';", \
                        "logs")
?CallResult.Success

?SubContainer.GetItem(ResultData, ContentType, "MyFile.jpg")
```

## ListItems

Lists the contents of the container as a CRLF() delimited string.

ListItems AS L (BYREF String as C [, SearchPrefix as C])

**Example**

```
dim ConnectionString    as C = "Provider='Disk';Container='C:\A5WebRoot';"
dim ResultString        as C
dim SubContainer        as A5Storage::SubContainer = null_value()
dim Item                as A5Storage::DataItem = null_value()

CallResult = A5Storage::SubContainer::Open(Container,  \
                        "Provider='Disk';Container='c:\A5Webroot';", \
                        "logs")
?CallResult.Success
?SubContainer.ListItems(ResultString)
Showvar(ResultString)
```

## Open

Opens (references) a storage container based on a connection string and the prefix provided.

```
STATIC Open AS CallResult (
        BYREF       SubContainer        AS A5Storage::SubContainer,
                    ConnectionStringIn  AS C
                    Prefix              AS C
                    [, Passphrase       AS C])

STATIC Open AS CallResult (
        BYREF       SubContainer        AS A5Storage::SubContainer,
                    Cache               AS A5Storage::NameCache,
                    ConnectionStringIn  AS C
                    Prefix              AS C
                    [, Passphrase       AS C])
```

**Examples**

```
dim CallResult          as CallResult
dim SubContainer        as A5Storage::SubContainer = null_value()

CallResult = A5Storage::SubContainer::Open(SubContainer,  \
                                "Provider='Disk';Container='c:\A5Webroot';", \
                                "logs")
if CallResult.Success
        …
```

**OR**

```
dim CallResult          as CallResult
dim SubContainer        as A5Storage::SubContainer = null_value()
dim Storages as C = <<%txt%
        Storage1        Provider='Disk';Container='C:\A5WebRoot1';
        Storage2        Provider='Disk';Container='C:\A5WebRoot2';
        %txt%
dim Cache as A5Storage::NameCache = new A5Storage::NameCache(Storages)


CallResult = A5Storage::SubContainer::Open(SubContainer,  Cache, \
                                "::NAME::MyConnectionString", \
                                "logs")
if CallResult.Success
        …
```

Note: The connection string passed in the second example does NOT have to be a named connection. If you DO pass a named connection, you must also provide a cache to look it up in.

## ReferenceItem

Creates and returns a reference to an item within the container.

Notes:
1. You must dimension Item before making the call.
2. No data is actually read until you ask for it.

```
ReferenceItem AS L (BYREF    Item          AS A5Storage::DataItem,
                             TargetPath    AS C)
```

**Example**

```
dim CallResult          as CallResult
dim SubContainer        as A5Storage::SubContainer = null_value()

CallResult = A5Storage::SubContainer::Open(SubContainer,  \
                                           "Provider='Disk';Container='c:\A5Webroot';", \
                                           "logs")
if CallResult.Success
        Dim Item A5Storage::DataItem = null_value()
        SubContainer.ReferenceItem(Item, "MyObjectName")
        …
```

## SaveItemToFile

SaveItemToFile AS L (SourcePath AS C, TargetPath AS C)

**Example**

```
dim CallResult          as CallResult
dim SubContainer        as A5Storage::SubContainer = null_value()

CallResult = A5Storage::SubContainer::Open(SubContainer,  \
                                           "Provider='Disk';Container='c:\A5Webroot';", \
                                           "logs")
if CallResult.Success
        ?SubContainer.SaveItemToFile("MyItempath/MyItem.jpg", \
                                           "c:\temp\MyItem.jpg")
        …
```

## SetItem

SetItem AS L (Source AS B, ContentType AS C, TargetPath AS C)
SetItem AS L (Source AS C, ContentType AS C, TargetPath AS C)
SetItem AS L (Source AS System::IO::Stream,
ContentType AS C, TargetPath AS C)

**Example**

```
dim CallResult          as CallResult
dim SubContainer        as A5Storage::SubContainer = null_value()

CallResult = A5Storage::SubContainer::Open(SubContainer,  \
                                           "Provider='Disk';Container='c:\A5Webroot';", \
                                           "logs")
if CallResult.Success
        ?SubContainer.SetItem("<html><body>Hello World</body> </html>",  \
                              "text/html",  \
                              "MyPath/ MyItem.html")
        …
```

## SetItemFromFile

```
SetItemFromFile AS L (    SourcePath AS C,
                          ContentType AS C,
                          TargetPath AS C
                          [, Offset as N = 0
                          [, Length as N = -1]])
```

**Example**

```
dim CallResult          as CallResult
dim SubContainer        as A5Storage::SubContainer = null_value()

CallResult = A5Storage::SubContainer::Open(SubContainer,  \
                                    "Provider='Disk';Container='c:\A5Webroot';", \
                                    "logs")
if CallResult.Success
        ?SubContainer.SetItemFromFile ("c:\temp\MyItem.jpg", \
                                    "MyItempath/MyItem.jpg")
        …
```

## WriteItemToStream

Retrieves an item from the container using the name you provide. The content type is also returned. The item is written to the stream you provide.

Note: You must dimension the resulting Stream, Length and ContentType variables prior to making the call.

```
WriteItemToStreamStream AS L (
                                Stream      AS System::IO::Stream
                        BYREF   Length      AS N,
                        BYREF   ContentType AS C,
                                TargetPath  AS C)
```

**Example**

```
dim Stream          as System::IO::Stream = null_value()
dim ContentType     as C
dim Length          as N
dim Buffer          as B
dim CallResult      as CallResult
dim SubContainer    as A5Storage::SubContainer = null_value()

Stream = new System::IO::MemoryStream()
CallResult = A5Storage::SubContainer::Open(SubContainer,  \
                                "Provider='Disk';Container='c:\A5Webroot';",
                                "logs")
if CallResult.Success
        if SubContainer.WriteItemToStream(Stream,  \
                                        Length, ContentType, "Speak.a5w")
                Stream.Seek(0, System::IO::SeekOrigin::Begin)

                dim Reader as System::IO::BinaryReader = \
                        new System::IO::BinaryReader(Stream)
                Buffer = Reader.ReadBytes(Length)
                Stream.close()
        else
                CallResult = SubContainer.CallResult
        end if
end if

if CallResult.Success
        showvar("Type: " + ContentType + crlf() + "Length: " + Length + crlf(2) + buffer,  \
                "Success")
else
        showvar(CallResult.Text, "Error")
end if
```

## *A5Storage::DataItem*

The A5Storage::DataItem class is essentially a reference to an object within the parent container. It does not hold any data, but has methods and properties that can be used to access the object itself and to get information about it without reading the object.

### Properties

| Property | Type | Writeable | Example Content |
|---|---|---|---|
| AbsolutePath | C | N | URL to reference the item directly in a browser:<br>For example:<br>file://c:\temp\teststorage\Hello<br>https://a5teststorage.blob.core.windows.net/text/Hello<br>https://s3.amazonaws.com/A5Test/Hello<br><br>Note: The drive path must be a UNC path or a common drive letter assigned to a share to use the value on a LAN. |
| CallResult | P | N | |
| ContentType | C | Yes | "text/html" |
| Exists | L | N | .t. |
| ModifiedTime | T | No | 09/29/2010 09:44:59 41 pm |
| Name | C | No | "speech/speak.html " |
| Size | N | No | 362 |

### CopyTo

This function will copy an item to another container. The function is optimized to use native methods to copy between the same types of containers when possible.

CopyTo AS L ( TargetContainer as A5Storage::DataContainer,
TargetItemName AS C)

**Example**

```
dim ConnectionString as C = "Provider='Disk';Container='C:\A5WebRoot';"
dim Container                as A5Storage::DataContainer = null_value()
dim TargetContainer          as A5Storage::DataContainer = null_value()

CallResult = A5Storage::DataContainer::Open(Container,  \
                    "Provider='Disk';Container='c:\A5Webroot';")
CallResult = A5Storage::DataContainer::Open(TargetContainer,  \
                    "Provider='Disk';Container='c:\A5Webroot2';")

Dim Item A5Storage::DataItem = null_value()
Container.ReferenceItem(Item, "MyObjectName")
Item.CopyTo(TargetContainer, “MyObjectName”)
```

## Delete

Delete the underlying object if it exists.  Returns false if an error occurs.

Note:  No error is returned if the object does not exist.

Delete AS L ()

**Example**

```
dim CallResult      as CallResult
dim Container       as A5Storage::DataContainer = null_value()

CallResult = A5Storage::DataContainer::Open(Container,  \
                                            "Provider='Disk';Container='c:\A5Webroot';")
if CallResult.Success
    Dim Item A5Storage::DataItem = null_value()
    If Container.ReferenceItem(Item, "MyObjectName")
        if .not. Item.Delete
            showvar(Item.CallResult.Text, "Error deleting Item")
    …
```

## Get

Get the data for the object. The data returned may be a blob or a string, depending on the type of the Data argument you pass.

Note: You must dimension both the Data and ContentTypeResult variables before calling this function.

Get AS L (BYREF Data AS B, BYREF ContentTypeResult AS C)
Get AS L (BYREF Data AS C, BYREF ContentTypeResult AS C)

**Example**

```
dim CallResult      as CallResult
dim Container       as A5Storage::DataContainer = null_value()

CallResult = A5Storage::DataContainer::Open(Container,  \
                                "Provider='Disk';Container='c:\A5Webroot';")
if CallResult.Success
    Dim Item A5Storage::DataItem = null_value()
    If Container.ReferenceItem(Item, "MyObjectName")

        Dim Data as B
        Dim ContentTypeResult as C
        if .not. Item.Get(Data, ContentTypeResult)
            showvar(Item.CallResult.Text, "Error in Get")
    …
```

## SaveToFile

Saves the data in an object to the file specified in TargetPath.

SaveToFile AS L (TargetPath AS C)

**Example**

```
dim CallResult      as CallResult
dim Container       as A5Storage::DataContainer = null_value()

CallResult = A5Storage::DataContainer::Open(Container,  \
                                "Provider='Disk';Container='c:\A5Webroot';")
if CallResult.Success
Dim Item A5Storage::DataItem = null_value()
    If Container.ReferenceItem(Item, "MyObjectName")
            if .not. Item.SaveToFile("c:\mydata.jpg)
```

## Set

Sets the object value and content type based on the values passed to the function.

Note: This function also accepts a stream. There may be some cases where a stream references the data from some other object of is the result of some other .NET function call. You can pass the stream as the data argument.

```
Set AS L (Source AS B, ContentTypeIn AS C)
Set AS L (Source AS C, ContentTypeIn AS C)
Set AS L (Source AS System::IO::Stream, ContentTypeIn AS C)
```

**Example**

```
dim CallResult        as CallResult
dim Container         as A5Storage::DataContainer = null_value()

CallResult = A5Storage::DataContainer::Open(Container,  \
                              "Provider='Disk';Container='c:\A5Webroot';")
if CallResult.Success
    Dim Item A5Storage::DataItem = null_value()
    If Container.ReferenceItem(Item, "MyObjectName")

        Dim Data                  as B = File.To_Blob("Myfile.jpg")
        Dim ContentTypeResult     as C =  "image/jpg"
        if .not. Item.Set(Data, ContentType)
            showvar(Item.CallResult.Text, "Error in Set")
    …
```

## SetFromFile

Set the object value from the file specified in SourcePath and sets the content based on ContentTypeIn.

```
SetFromFile AS L (  SourcePath AS C,  ContentTypeIn AS C
                    [, Offset as N = 0 [, Length as N = -1]])
```

**Example**

```
dim CallResult        as CallResult
dim Container         as A5Storage::DataContainer = null_value()

CallResult = A5Storage::DataContainer::Open(Container,  \
                              "Provider='Disk';Container='c:\A5Webroot';")
if CallResult.Success
    Dim Item A5Storage::DataItem = null_value()
    If Container.ReferenceItem(Item, "MyObjectName")
        if .not. Item.SetFromFile(File.To_Blob("MyFile.jpg", "image/jpg")
            showvar(Item.CallResult.Text, "Error in SetFromFile")
    …
```

## WriteToStream

Retrieves the item data and writes it to a System::IO::Stream object. The content type and length written are also returned.

Note: You must dimension the resulting Stream and ContentType and Length variables prior to making the call.

```
WriteToStream AS L (
                        Stream          AS System::IO::Stream,
                BYREF LengthWritten     AS N,
                BYREF ContentType       AS C)
```

**Example**

```
dim Stream              as System::IO::Stream = null_value()
dim ContentType as C
dim Length              as N
dim Buffer              as B
dim CallResult  as CallResult
dim Container   as A5Storage::DataContainer = null_value()
dim Item                as A5Storage::DataItem = null_value()

Stream = new System::IO::MemoryStream()
CallResult = A5Storage::DataContainer::Open(Container,
"Provider='Disk';Container='c:\A5Webroot';")
if CallResult.Success
        if Container.ReferenceItem(Item, "Speak.a5w")
                if Item.WriteToStream(Stream, Length, ContentType)
                        Stream.Seek(0, System::IO::SeekOrigin::Begin)
                        dim Reader as System::IO::BinaryReader = new \
                                        System::IO::BinaryReader(Stream)
                        Buffer = Reader.ReadBytes(Length)
                        Stream.close()
                else
                        CallResult = Item.CallResult
                end if
        else
                CallResult = Container.CallResult
        end if
end if

if CallResult.Success
        showvar("Type: " + ContentType + crlf() + "Length: " + Length + crlf(2) + buffer,
"Success")
else
        showvar(CallResult.Text, "Error")
end if
```

## *A5Storage::NameCache*

The A5Storage::NameCache is a container for named connection strings. The caller is responsible for managing the contents and for providing the cache to use when calling A5Storage::DataContainer::Open().

### Properties

| Property | Type | Writeable | Description |
|---|---|---|---|
| CallResult | CallResult | N | Contains the status of the last call made on the instance. |
| Names | C | Yes | Gets or sets the entire cache from or to a string of the format:<br><br>"name<tab>connectionstring<crlf><br>… "<br><br>Each row contains a tab separated name and connection string pair. The rows are delimited with a carriage return and line feed (crlf). |
| NamedConnectionPrefix | C | N | "::NAME::" |

### Constructors

A5Storage::NameCache()
A5Storage::NameCache(Storages as C)

**Example 1**

```
dim Storages as C = <<%txt%
        Storage1        Provider='Disk';Container='C:\A5WebRoot1';
        Storage2        Provider='Disk';Container='C:\A5WebRoot2';
        %txt%
dim Cache as A5Storage::NameCache
if Cache.AddItem(Storages)
        …
else
        ShowVar("Oops!" + crlf() + Cache.CallResult.Text)
end if
```

**Example 2**

```
dim Storages as C = <<%txt%
        Storage1        Provider='Disk';Container='C:\A5WebRoot1';
        Storage2        Provider='Disk';Container='C:\A5WebRoot2';
        %txt%
dim Cache as A5Storage::NameCache = new A5Storage::NameCache(Storages)
```

## AddItems

AddItem AS L (  NamesandValues  AS C)

**Example**

```
Dim Cache as A5Storage::NameCache
dim Storages as C = <<%txt%
        Storage1        Provider='Disk';Container='C:\A5WebRoot1';
        Storage2        Provider='Disk';Container='C:\A5WebRoot2';
        Invalid         Provider='Disk';Container='AA:\abc';
        %txt%
if Cache.AddItem(Storages)
        …
else
        ShowVar("Oops!" + crlf() + Cache.CallResult.Text)
end if
```

## Clear

Clear AS L ()

**Example**

```
Dim Cache as A5Storage::NameCache
Cache.Clear()
```

## Find

Find AS L (Key as C, BYREF Value as C)

**Example**

```
dim Cache               as A5Storage::NameCache
dim ConnectionString    as C
if Cache.Find("MyName", ConnectionString)
        …
else
        ShowVar("Oops!" + crlf() + Cache.CallResult.Text)
end if
```

## Set

Adds or replaces a connection string in the cache.

```
Set AS L (    Name              AS C,
              ConnectionString  AS C)
```

**Example**

```
Dim Cache as A5Storage::NameCache
Cache.Set("MyName", "MyConnectionString"
```

## Using A5Storage::NameCache with Containers

The example script below shows how you can create and use a cache of names with A5Storage::Container to automatically translate named connections.

```
dim Storages as C = <<%txt%
Storage1        Provider='Disk';Container='C:\A5WebRoot1';
Storage2        Provider='Disk';Container='C:\A5WebRoot2';
Invalid         Provider='Disk';Container='AA:\abc';
%txt%

dim TestStorages as C = <<%txt%
Storage1
Storage2
Invalid
NotFound
%txt%

Result = RunTest(Storages, TestStorages)
showvar(Result)



FUNCTION RunTest as C (Storages as C, TestStorages as C)
dim Result as C
dim cache as a5storage::namecache = new a5storage::namecache(Storages)

for i = 1 to w_count(TestStorages, crlf())
        CurrentItem = word(TestStorages, i, crlf())
        CurrentName = word(CurrentItem, 1, chr(9))
        Result = Result + "Testing Name: " + CurrentName + crlf() + TestName(Cache, "::name::" +
CurrentName)
next

RunTest = Result
END FUNCTION


FUNCTION TestName(Cache as A5Storage::NameCache, Name as C)

dim Result as C

dim sc as a5storage::datacontainer = null_value()

CallResult = A5Storage::DataContainer::Open(sc, Cache, Name)
if CallResult.Success
        dim ItemName            as C = "test/Data.Txt"
        dim ItemValue           as C = "StorageData"
        dim ItemContentType as C = "text"

        if sc.SetItem(ItemValue, ItemContentType, ItemName)

                dim di as A5Storage::DataItem = null_value()

                if sc.ReferenceItem(di, ItemName)
                        dim Text AS C
                        dim ContentType as C
```

```
                        di.Get(Text, ContentType)
                        Result = "Item: " + ItemName + " = " + Text + crlf()
                        Result = Result + chr(9) + chr(9) + "Modified:                " + di.ModifiedTime + crlf()
                        Result = Result + chr(9) + chr(9) + "ContentType: " + di.ContentType + crlf()
                        Result = Result + chr(9) + chr(9) + "Size:        " + di.Size + crlf(2)
                else
                        Result = "Error referencing item '" + ItemName + "':" + sc.CallResult.Text
                end if
        else
                Result = "Error setting item: " + sc.CallResult.Text + crlf()
        end if
else
        Result = "Error opening container '" + Name + ": " + CallResult.Text + crlf()
end if
TestName = Result
END FUNCTION
```

# Using Storages with Uploaded Files

The class INET::UploadedFile now has a method to allow saving an uploaded file directly to a storage. This method bypasses memory altogether and saves the file from disk to the storage using buffered reads and writes.

## *INET::UploadedFile::SaveToStorage*

```
INET::UploadedFile::SaveToStorage as L (  DestinationStorage    as P,
                                          TargetMember          as C,
                                          ContentType           as C)
```

Note: DestinationStorage can be either an instance of A5Storage::DataContainer or an instance of A5Storage::SubContainer.

### Basic Example

```
if eval_valid("Request.Variables.File1")

        DiskContainerName       = "C:\Temp\Disk"
        DiskConnectionString    = "Provider='Disk';Container='" + DiskContainerName + "';"

        File                    = Request.Variables.File1
        FileName                = File.FileName
        FileSize                = File.Size
        FileContentType         = File.ContentType

        ? "File uploaded: " + FileName + " (" + FileSize + " bytes)<br/>"

        dim Storage as A5Storage::SubContainer = null_value()

        CallResult = A5Storage::SubContainer::Open(Storage, DiskConnectionString, "Test1")
        if CallResult.Success
                if File.SaveToStorage(Storage, FileName, FileContentType)
                        ? "File written to storage: " + FileName + " (" + FileSize + " bytes in " + \
                else
                        ? "File write to storage failed: " + File.CallResult.Text + "<br/>"
                end if
        else
                ? "File upload failed.  Unable to open storage." + CallResult.Text + "<br/>"
        end if
end if
```

## *A Complete Example of an A5W Page*

```
<html>
<head><title>File Upload</title></head>
<body>
<%a5
if eval_valid("Request.Variables.File1")

	DiskContainerName		= "C:\Temp\Disk"
	DiskConnectionString		= "Provider='Disk';Container='" + DiskContainerName + "';"

	AzureAccountName		= "a5webservertest"
	AzureAccessKey			= "<your access key here"
	AzureContainerName		= "a5webservertest"
	AzureConnectionString		= "Provider='Azure';Account='" + AzureAccountName + \
		"';AccessKey='" + AzureAccessKey + "';Container='" + AzureContainerName +"';"

	AzureTestContainerName		= "a5webservertest"
	AzureTestConnectionString	=
			\"Provider='Azure';UseDevelopmentStorage='true';Container='"\
			 + AzureContainerName +"';"

	AmazonAccessKey				= "<your access key>"
	AmazonSecretKey				= "<your amazon secret key>"
	AmazonContainerName		= "A5WebServerTest-Application"
	AmazonConnectionString		= "Provider='AmazonS3';AccessKey='" \
			+ AmazonAccessKey + "';SecretKey='" \
			+ AmazonSecretKey + "';Container='" + AmazonContainerName +"';"

	dim ConnectionStrings[4] as C = [ DiskConnectionString, AzureConnectionString,  \
					AzureTestConnectionString, AmazonConnectionString ]

	dim Timer as Util::Timer
	File		= Request.Variables.File1
	FileName	= File.FileName
	FileSize	= File.Size
	FileContentType	= File.ContentType

	? "File uploaded: " + FileName + " (" + FileSize + " bytes)<br/><br/>"

	dim Storage as A5Storage::SubContainer = null_value()
	for i = 1 to ConnectionStrings.Size()
		ConnectionString = ConnectionStrings[i]
		? "Opening connection string at index " + i + " - " + ConnectionString + "." + "<br/>"


		Storage = null_value()

		CallResult = A5Storage::SubContainer::Open(Storage, ConnectionString, "Test1")
		if CallResult.Success
			dim UploadTimer as Util::Timer
			UploadTimer.Start()

			if File.SaveToStorage(Storage, FileName, FileContentType)
				? "File written to storage: " + FileName + " (" + FileSize + " bytes in " \
					+ UploadTimer.ElapsedMilliseconds + " milliseconds.)<br/>"
			else
				? "File write to storage failed: " + File.CallResult.Text + "<br/>"
			end if
		else
```

```
                        ? "File upload failed.  Unable to open storage." + CallResult.Text + "<br/>"
                    end if
                    ? "<br/>"
                next
                Response.Write("<hr/>")
                ? "Total time to save: " + Timer.ElapsedMilliseconds + " milliseconds." + "<br/>"
                Response.Write("<hr/>")

end if
%>

<form action="<%a5 ?Request.ScriptName%>" method="post" enctype="multipart/form-data">
<input type="file" name="File1"/><br/>
<input type="submit" name="cmd" value="Upload File"/>
</form>

</body>
</html>
```