MV Share Store API (Version: 1.4)

Types of API

There are basically two types of API:

1. The normal API using instances of classes such as DataItem, DataPack, DataShare & DataStore.
2. The static API implemented in the **Data class** which provides a gateway into the underlying classes above.

In every case this document will detail the actual property or function definition and it's IntelliSense comments.

Conventions

- Static Functions in classes like Data and ShareStore are generally capitalised (such as CreateOrOpenShareStore).
- Non static **functions** in classes such as DataPack, DataShare and DataStore are sometimes camelCase (such as addDataItems) but can also be more standardised (such as AddRange).
- Defines, Types, Structs and Enums are all upper case (such as DataType.TYPE).
- Properties are generally capitalised and named after the Class or Field they represent (such as DataPack or DataShare).
- Many functions have default values defined for some of their parameters (see the IntelliSense comments for these).

All public functions and properties should be fully commented so their use, parameters and return values should be displayed by IntelliSense.

Structure

The MVShareStore library has a very hierarchical structure:

ShareStore

A collection of all the posted / published DataShares. Multiple ShareStores are allowed on the same host. Each ShareStore has a default capacity for over 5,000 DataShares but this can be increased if required.

DataShare

A DataPack that can be named, have an owner and posted / published into a ShareStore for use by other processes.

DataStore

A DataPack that can be saved to a file on disk and opened again later.

DataPack

A collection of DataItems that can optionally be named and accessed using their names or their index into the DataPack. DataItems can be added to, deleted from or inserted into a DataPack at any time. DataPacks can also be DataItems enabling them to be nested do any depth desired.

Note that in this release the API exposed by DataPacks has changed to bring it more in line with the standard .NET Collections API. The legacy API is still there and can be used using a different build of the MVShareStore Library (included in the release).

DataItem

DataItems each have a DataType, a value and optionally a name and can be accessed using their name or index in the DataPack. Duplicate DataItem names in a DataPack are not allowed.

Data Class Static API

The Data class exposes almost all the library's functionality through a set of static functions and properties, meaning that an instance of this class should never need to be instantiated. It should be noted that many of these functions have default parameters defined, hopefully making their use easier and more flexible.

It also implements the concept of a currently active DataPack, currently active DataShare and currently active DataStore which can be used by default.

It also contains a property called LastError which should contain a description of the last error that the library reported meaning that if a function returns an error result the reason for this should be in the LastError string. A history of errors that have happened can also be enabled and viewed.

Properties

/// The Version of the Library as a string
public static string Version

/// A holding place for any information you want to put in here.
public static string Info

/// A description of the last error that occurred
public static string LastError

/// A List of all the errors that have been put into the error message history list
public static List<string> Errors

/// The number of error messages in the error history list
public static int ErrorCount

/// The last path / directory that was used to save DataStores to
public static string LastPath

/// The name of the last file opened or saved
public static string LastFile

/// The Date Format to use when displaying dates and times (in C# format)
public static string DateFormat

/// The current Owner / User / Group to be used for DataShares
public static string CurrentOwnerUser

/// The currently selected / active Data Pack.
public static DataPack CurrentDataPack

/// The currently selected / active Data Share.
public static DataShare CurrentDataShare

/// The currently selected / active Data Store.
public static DataStore CurrentDataStore

Functions

The following are all the public static functions provided by the Data Class API:

ShareStore and Settings Functions

/// Create a new or open an existing Share Store optionally using the specified name and maximum lengths for the DataStore Name and Owner fields
/// < "name">The Name of the ShareStore to open (if null it will use the default Name)</param>
/// < "maxNameLen">The (optional) maximum length that DataShare Names can be</param>
/// < "maxOwnerLen">The (optional) maximum length that DataShare Owners can be</param>
/// <returns>True if the Share Store Global Area was successfully opened otherwise false if it was already open or there was a problem</returns>
public static bool CreateOrOpenShareStore(string name = null, int maxNameLen = 0, int maxOwnerLen = 0)

/// Open an existing Share Store optionally using the specified or default name
/// < "name">The Name of the ShareStore to open (if null it will use the default Name)</param>
/// <returns>True if the Share Store Global Area was successfully opened otherwise false if it was already open or there was a problem</returns>
public static bool OpenShareStore(string name = null)

/// Close the currently open Share Store
public static void CloseShareStore()

/// Get all the Data Shares in the Share List for this Share Store
/// <“filters">Optionally set filters regarding what is returned in the Data Share Header List</param>
/// <“sortBy">Optionally specify the field you want to sort the Share List by and whether it should be in ascending or descending order</param>
/// <returns>All the Headers in the Global Header List optionally filtered and / or sorted or an empty list if there weren't any or null if there was a problem</returns>
public static List<Header> Shares(Filters filters = null, Sort sortBy = null)

/// Close all the open DataPacks, DataShares and DataStores (optionally saving those if required). This should be called when shutting the program using the library down.
/// < "save">Optionally save any DataStores that have been changed but not yet saved (defaults to true)</param>
public static void CloseAll(bool save = true)

/// Read the Library Settings from the DataStore on disk. If it is not there the default settings will be used
/// <returns>True if successful otherwise false</returns>
public static bool ReadSettings()

/// Save the current Library Settings to a DataStore on disk. Create a new one if it does not already exist

/// <returns>True if successful otherwise false</returns>
public static bool SaveSettings()

/// Check to see if the filename passed is in a valid format, optionally checking to also make sure it exists
/// < "filename">The filename to check for validity</param>
/// < "mustExist">Must a file with that name also exist (defaults to false)</param>
/// <returns>True if the filename passed is a valid filename and optionally also if it exists otherwise false</returns>
public static bool isValidFilename(string filename, bool mustExist = false)

DataItem Functions

/// Static function used to create a new DataItem based on the specified type and using the value in the string input (which will be validated)
/// <"type">The Type of the DataItem to be created</param>
/// <"value">The value to put into the DataItem in a string which will be validated</param>
/// <"name">The (optional) name of the variable (defaults to null if not supplied)</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or null if there was a problem (error in Data.LastError)</returns>
public static DataItem CreateDataItem(DataType.TYPE type, string value, string name = null, string code = null)

/// Static function used to create a new DataItem based on the specified type and using the value in the object input (which will be validated)
/// < "type">The Type of the DataItem to be created</param>
/// < "value">The value to put into the DataItem in an object which will be validated</param>
/// < "name">The (optional) name of the variable (defaults to null if not supplied)</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or null if there was a problem (error in Data.LastError)</returns>
public static DataItem CreateDataItem(DataType.TYPE type, object value, string name = null, string code = null)

/// Create and return a new Bool Data Item
/// <”value">The string containing the value to assign to the Bool Data Item (defaults to false if not supplied or invalid)</param>
/// <”name">The (optional) name of the variable</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or DataItem with the default value (false) if there was a problem</returns>
public static DataItem CreateBool(string value, string name = null, string code = null)

/// Create and return a new Byte Data Item
/// <"value">The string containing the value to assign to the Byte Data Item (defaults to 0 if not supplied or invalid)</param>

/// <”name">The (optional) name of the variable</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or DataItem with the default value (0) if there was a problem</returns>
public static DataItem CreateByte(string value, string name = null, string code = null)

/// Create and return a new Char Data Item
/// <"value">The string containing the value to assign to the Char Data Item (defaults to ' ' if not supplied or invalid)</param>
/// <” name">The (optional) name of the variable</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or DataItem with the default value (' ') if there was a problem</returns>
public static DataItem CreateChar(string value, string name = null, string code = null)

/// Create and return a new Short Data Item
/// <”value">The string containing the value to assign to the Short Data Item (defaults to 0 if not supplied or invalid)</param>
/// <”name">The (optional) name of the variable</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or DataItem with the default value (0) if there was a problem</returns>
public static DataItem CreateShort(string value, string name = null, string code = null

/// Create and return a new Int Data Item
/// <”value">The string containing the value to assign to the Int Data Item (defaults to 0 if not supplied or invalid)</param>
/// <”name">The (optional) name of the variable</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or DataItem with the default value (0) if there was a problem</returns>
public static DataItem CreateInt(string value, string name = null, string code = null, string code = null)

/// Create and return a new Long Data Item
/// <”value">The string containing the value to assign to the Long Data Item (defaults to 0 if not supplied or invalid)</param>
/// <”name">The (optional) name of the variable</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or DataItem with the default value (0) if there was a problem</returns>
public static DataItem CreateLong(string value, string name = null, string code = null)

/// Create and return a new Float Data Item
/// <”value">The string containing the value to assign to the Float Data Item (defaults to 0.0 if not supplied or invalid)</param>

/// <”name">The (optional) name of the variable</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or DataItem with the default value (0.0) if there was a problem</returns>
public static DataItem CreateFloat(string value, string name = null, string code = null) {

/// Create and return a new Double Data Item
/// <”value">The string containing the value to assign to the Double Data Item (defaults to 0.0 if not supplied or invalid)</param>
/// <”name">The (optional) name of the variable</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or DataItem with the default value (0.0) if there was a problem</returns>
public static DataItem CreateDouble(string value, string name = null, string code = null)

/// Create and return a new Decimal Data Item
/// <”value">The string containing the value to assign to the Decimal Data Item (defaults to 0.0 if not supplied or invalid)</param>
/// <”name">The (optional) name of the variable</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or DataItem with the default value (0.0) if there was a problem</returns>
public static DataItem CreateDecimal(string value, string name = null, string code = null)

/// Create and return a new String Data Item
/// <”value">The string value to assign to the String Data Item (defaults to null if not supplied)</param>
/// <”name">The (optional) name of the variable</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or DataItem with null if there was a problem</returns>
public static DataItem CreateString(string value, string name = null, string code = null)

/// Create and return a new Datapack Data Item
/// < "value">The DataPack object containing the value to assign to the Datapack Data Item (defaults to null if not supplied or invalid)</param>
/// < "name">The (optional) name of the variable</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or DataItem with the default value of null if there was a problem</returns>
public static DataItem CreateDatapack(DataPack value, string name = null, string code = null)

DataPack Functions

Constructors

/// Create a new DataPack, (optionally showing a form to add Data Items to the DataPack) and make it the current DataPack.
/// <”populate">Show the form to add new DataItems to populate the DataPack with</param>
/// <”capacity">The optional expected capacity of the DataPack</param>
/// <returns>The new DataPack just created or null if it fails or there is a problem (Error will be in LastError)</returns>
public static DataPack CreateDataPack(bool populate = false, int capacity = 0)

/// Create a new DataPack and populate it using the supplied DataPack (basically cloning it by making it a 'deep copy').
/// <”dataItems">The DataPack to use to populate the new DataPack</param>
/// <”editItems">Edit the Data Items in the new DataPack using a simple form after it has been created</param>
/// <returns>The new DataPack populated with the Data Items in the DataPack supplied</returns>
public static DataPack CreateDataPack(DataPack dataPack, bool editItems = false)

/// Create a new DataPack and populate it using the supplied List of DataItems.
/// <”dataItems">The List of DataItems to use to populate the DataPack</param>
/// <”editItems">Edit the Data Items in the new DataPack using a simple form after it has been created</param>
/// <returns>A new DataPack populated with the Data Items in the List supplied</returns>
public static DataPack CreateDataPack(List<DataItem> dataItems, bool editItems = false)

/// Create a new DataPack and populate it using the supplied array of DataItems.
/// <”dataItems">The array of DataItems to use to populate the DataPack</param>
/// <”editItems">Edit the Data Items in the new DataPack using a simple form after it has been created</param>
/// <returns>A new DataPack populated with the Data Items in the array supplied</returns>
public static DataPack CreateDataPack(DataItem[] dataItems, bool editItems = false)

General Functions

/// Show a simple form to edit Data Items in the supplied DataPack (which is then made the currently active DataPack). If this is null then the currently active DataPack will be edited instead.
/// <”dataPack">The DataPack to edit. If it is null then the currently active DataPack will be edited instead. If the editing is cancelled out of the DataPack copy will be returned unchanged</param>
/// <returns>The edited (and now currently active) DataPack or the original unchanged if the editing was cancelled</returns>
public static DataPack EditDataPack(DataPack dataPack = null)

/// Show the all the DataItems in the DataPack in a simple form. If the supplied DataPack is null then the currently active DataPack will be shown instead
/// <“dataPack">The Data Pack to show the contents of. If it is null then the currently active DataPack will be shown instead</param>
public static void ListDataPack(DataPack dataPack = null)

/// Duplicate the supplied DataPack and return a 'deep copy' which will become the currently active DataPack.
/// <”dataPack">The DataPack to duplicate in this DataPack</param>
/// <”editItems">Optionally use a simple form to edit the Data Items after making the copy (defaults to no). If the editing is cancelled out of the DataPack copy will be returned unchanged</param>
/// <returns>A new DataPack populated with the Data Items in the List supplied</returns>
public static DataPack DuplicateDataPack(DataPack dataPack, bool editItems = false)

/// Save either the supplied or the currently active DataPack to disk, optionally supplying the file name. Note that any existing file will be automatically overwritten
/// <”dataPack">The DataPack to save to disk. If it is null the currently active DataPack will be saved instead</param>
/// <”filename">An optional name for the file. If not supplied then either the last file name used (if available) will be used or a new file name will be created / determined</param>
/// <returns>True if the DataPack was successfully saved to disk otherwise false (with the problem in Data.LastError)</returns>
public static bool SaveDataPack(DataPack dataPack = null, string filename = null)

/// Close the supplied DataPack and invalidate it. If no DataPack supplied then the currently active pack will be closed instead
/// <“dataPack">The DataPack to close / invalidate / delete or null to close / invalidate / delete the currently active Data Pack</param>
public static void CloseDataPack(DataPack dataPack = null)

Convert DataPacks

/// Convert the DataPack supplied into a new DataShare. If it is null then the currently active Data Pack will be used
/// <”dataPack">The DataPack to use to populate the DataShare. If null it will convert the currently active DataPack</param>
/// <”showCreateDialog">Optionally show the create DataShare dialog form to enter additional information</param>
/// <”name">The optional Name of the Data Share (not required if showDialog is set to true)</param>
/// <”owner">The Owner of the Data Share. This is required for private Data Stores (not required if showDialog is set to true)</param>
/// <”readOnly">Is this Data Share read only after having been posted? (not required if showDialog is set to true)</param>
/// <”priv">Is this Data Share private (restricted to it's Owner) or not? Defaults to public (not required if showDialog is set to true)</param>
/// <returns>The (currently active) DataShare just created or null if there was a problem (Error will be in LastError)</returns>

public static DataShare ConvertDataPackToDataShare(DataPack dataPack = null, bool showCreateDialog = false, string name = null, string owner = null, bool readOnly = false, bool priv = false)

/// Convert the DataPack supplied into a new DataStore. If it is null then the currently active Data Pack will be used
/// <”dataPack">The DataPack to use to populate the DataStore. If null it will convert the currently active DataPack</param>
/// <”showCreateDialog">Optionally show the create DataStore dialog form to enter additional information</param>
/// <”name">The name of the Data Store (which may be used to help define the file name) (not required if showDialog is set to true)</param>
/// <”path">The optional path / directory where you want this Data Store to be saved to / read from (not required if showDialog is set to true)</param>
/// <returns>The (currently active) DataStore just created or null if there was a problem (Error will be in LastError)</returns>
public static DataStore ConvertDataPackToDataStore(DataPack dataPack = null, bool showCreateDialog = false, string name = null, string path = null)

Add Data Items

/// Add another DataItem to the list of data items stored in the DataPack and return it's index in the list
/// <”dataItem">The DataItem to add to the DataPack</param>
/// <returns>The index of the item in the DataPack (or -1 if there was a problem)</returns>
public static int AddDataPackItem(DataItem dataItem)

/// Add another DataItem to the list of data items stored in the DataPack and return it's index in the list
/// <”type">The Type of the DataItem to be added</param>
/// <”value">The value to put into the DataItem in a string which will be validated</param>
/// <”name">The (optional) name of the DataItem (defaults to null if not supplied)</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The index of the item in the DataPack (or -1 if there was a problem)</returns>
public int AddDataPackItem(DataType.TYPE type, string value, string name = null, string code = null)

/// Add another DataItem to the list of data items stored in the DataPack and return it's index in the list
/// < "type">The Type of the DataItem to be added</param>
/// < "value">The value to put into the DataItem in an object which will be validated</param>
/// < "name">The (optional) name of the DataItem (defaults to null if not supplied)</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The index of the item in the DataPack (or -1 if there was a problem)</returns>
public int AddDataPackItem(DataType.TYPE type, object value, string name = null, string code = null)

/// Add additional DataItems to the list of data items stored in the DataPack and the number added

/// <”dataPack">The DataPack to add the Items from to the DataPack</param>
/// <returns>The number of new Items added</returns>
public static int AddDataPackItems(DataPack dataPack)

/// Add additional DataItems from a List of Data Items to the Data Items already stored in this DataPack and return the number added
/// <”dataItems">The List of Data Items to add to the DataPack</param>
/// <returns>The number of new Items added</returns>
public static int AddDataPackItems(List<DataItem> dataItems)

/// Add additional DataItems from an array of Data Items to the list of Data Items already stored in this DataPack and return the number added
/// <”dataItems">The array of Data Items to add to the DataPack</param>
/// <returns>The number of new Items added</returns>
public static int AddDataPackItems(DataItem [] dataItems)

Edit Data Items

/// Insert the supplied Data Item at the specified index in the current DataPack
/// <”index">The index in the DataPack to start the insertion of Data Items from (if an item at that index does not currently exist then it will be added instead)</param>
/// <”dataItem">The DataItem to insert into the DataPack</param>
/// <returns>The index of the inserted DataItem or -1 if there was a problem</returns>
public static int InsertDataPackItem(int index, DataItem dataItem)

/// Insert additional DataItems from another DataPack into the list of Data Items stored in this DataPack at the specified index and return the number added
/// <”index">The index in the DataPack to start the insertion of Data Items from (if an item at that index does not currently exist then it will be added instead)</param>
/// <“dataPack">The DataPack to insert the Items from into the current DataPack</param>
/// <returns>The number of new Items added</returns>
public int InsertDataPackItems(int index, DataPack dataPack)

/// Insert additional DataItems from a List of DataItems into the list of Data Items stored in this DataPack at the specified index and return the number added
/// <”index">The index in the DataPack to start the insertion of Data Items from (if an item at that index does not currently exist then it will be added instead)</param>
/// <”dataItems">The List of Data Items to insert the Items from into the current DataPack</param>
/// <returns>The number of new Items added</returns>
public int InsertDataPackItems(int index, List<DataItem> dataItems)

/// Insert additional DataItems from an array of DataItems into the list of Data Items stored in this DataPack at the specified index and return the number added
/// <”index">The index in the DataPack to start the insertion of Data Items from (if an item at that index does not currently exist then it will be added instead)</param>
/// <”dataItems">The array of Data Items to insert the Items from into the current DataPack</param>
/// <returns>The number of new Items added</returns>
public int InsertDataPackItems(int index, DataItem [] dataItems)

/// Edit the Data Item at the specified index in the current DataPack replacing it with the supplied Data Item
/// < "index">The index in the DataPack of the Data Item to change</param>
/// <”dataItem">The Data Item to replace the existing Data Item with</param>
/// <returns>The replaced Data Item or null if there was a problem</returns>
public DataItem EditDataPackItem(int index, DataItem dataItem)

/// Edit the Data Item identified by name in the current DataPack replacing it with the supplied Data Item
/// < "name">The original name of the Data Item to change (to enable it to be found)</param>
/// < "dataItem">The Data Item to replace the existing Data Item with</param>
/// <returns>The replaced Data Item or null if there was a problem</returns>
public DataItem EditDataPackItem(string name, DataItem dataItem)

/// Delete the Data Item at the specified index into the currently active DataPack
/// <”index">The index in the current DataPack of the Data Item to delete</param>
/// <returns>The number of Data Items remaining or -1 if there was a problem</returns>
public static int DeleteDataPackItem(int index)

/// Delete the Data Item at the specified index into the currently active DataPack
/// < "name">The name of the Data Item to delete in the current DataPack</param>
/// <returns>The number of Data Items remaining or -1 if there was a problem</returns>
public static int DeleteDataPackItem(string name)

Merge, Extract and Sort Functions

/// Merge the Data Items in 2 DataPacks together and return the resulting DataPack
/// <”dataPack1">The DataPack to have the second DataPack merged into</param>
/// <”dataPack2">The DataPack to merge into the first DataPack</param>
/// <returns>The resulting merged DataPack or null if there was a problem</returns>
public static DataPack MergeDataPackItems(DataPack dataPack1, DataPack dataPack2)

/// Extract and optionally remove a DataPack DataItem from this DataPack
/// < "name">The name of the DataPack DataItem to extract. Use the Path delimiter '/' to access an embedded / nested DataPack</param>
/// <”remove">Optionally remove the DataPack DataItem from the parent DataPack (defaults to false)</param>
/// <returns>The extracted DataPack or null if it could not be found</returns>
public static DataPack ExtractDataPack(string name, bool remove = false)

/// Extract and optionally remove a DataPack DataItem from this DataPack
/// <"dataPack">The DataPack to extract from (it will default to the currently active DataPack if not supplied)</param>
/// <”name">The name of the DataPack DataItem to extract. Use the Path delimiter '/' to access an embedded / nested DataPack</param>
/// <”remove">Optionally remove the DataPack DataItem from the parent DataPack (defaults to false)</param>
/// <returns>The extracted DataPack or null if it could not be found</returns>
public static DataPack ExtractDataPack(DataPack dataPack, string name, bool remove = false)

/// Sort the currently active DataPack and return it in sorted ascending or descending Data Item Name order (unnamed Data Items will all be at the beginning for ascending sorts and at the end for descending sorts)
/// < "descending">Should the sort order be descending (defaults to ascending)</param>
/// < "sortByCodeFirst">Sort the Data Items by the Code Field first before sorting my Name (defaults to false)</param>
/// <returns>The number of DataItems sorted if it was successful otherwise -1 if there was a problem</returns>
public static int SortDataPackItems(bool descending = false, bool sortByCodeFirst = false)

/// Sort the DataPack passed and return it in sorted ascending or descending Data Item Name order (unnamed Data Items will all be at the beginning for ascending sorts and at the end for descending sorts)
/// <"dataPack">The DataPack to sort and return</param>
/// <"descending">Should the sort order be descending (defaults to ascending)</param>
/// < "sortByCodeFirst">Sort the Data Items by the Code Field first before sorting my Name (defaults to false)</param>/// <returns>The sorted DataPack or null if there was a problem</returns>
public static DataPack SortDataPackItems(DataPack dataPack = null, bool descending = false, bool sortByCodeFirst = false)

Get and Find Functions

/// Get the Data Item at the specified index in the currently active or supplied DataPack
/// <"index">The index of the Data Item required (must be greater than -1 and less than the Item Count)</param>
/// <"dataPack">The optional DataPack to get the Data Item from or null to use the currently active DataPack</param>
/// <returns>The DataItem at the specified index or null if there was a problem (a description of which will be in Data.LastError)</returns>
public static DataItem getDataPackItem(int index, DataPack dataPack = null)

/// Get the Data Item with the specified name in the currently actibe or supplied DataPack
/// <“name">The name of the Data Item required</param>
/// <“dataPack">The optional DataPack to get the Data Item from or null to use the currently active DataPack</param>
/// <returns>The DataItem with the specified name or null if there was a problem (a description of which will be in Data.LastError)</returns>
public static DataItem getDataPackItem(string name, DataPack dataPack = null)

/// Find the Data Item at the specified index in this DataPack
/// <"index">The index of the Data Item required (must be greater than -1 and less than the Item Count)</param>
/// <"dataPack">The optional DataPack to get the Data Item from or null to use the currently active DataPack</param>
/// <returns>The DataItem at the specified index or null if there was a problem (a description of which will be in Data.LastError)</returns>
public static DataItem findDataPackItem(int index, DataPack dataPack = null)

/// Find the Data Item with the specified name in this DataPack

/// < "name">The name of the Data Item required</param>
/// < "dataPack">The optional DataPack to get the Data Item from or null to use the currently active DataPack</param>
/// <returns>The DataItem with the specified name or null if there was a problem (a description of which will be in Data.LastError)</returns>
public static DataItem findDataPackItem(string name, DataPack dataPack = null)

DataShare Functions

Constructors

/// Show the create DataShare dialog and use it to create a new DataShare, make it the current DataShare and store it in the internal list of Data Stores.
/// <returns>The new DataShare just created or null if it fails or there is a problem (Error will be in LastError)</returns>
public static DataShare CreateDataShare()

/// Show the create DataShare dialog and use it to create a new DataShare using the supplied DataPack / DataShare / DataStore, make it the current DataShare and store it in the internal list of Data Shares.
/// < "dataPack">The DataPack / DataShare / DataStore which will be used to populate the Data Items in the DataShare</param>
/// <returns>The new DataShare just created or null if it fails or there is a problem (Error will be in LastError)</returns>
public static DataShare CreateDataShare(DataPack dataPack)

/// Create a new Data Share of the specified type and store it in the internal list of Data Stores.
/// < "name">The optional Name of the Data Share</param>
/// < "owner">The Owner of the Data Share. This is required for private Data Stores</param>
/// < "readOnly">Is this Data Share read only after having been posted?</param>
/// < "priv">Is this Data Share private (restricted to it's Owner) or not? Defaults to public</param>
/// <returns>The new DataShare just created or null if it fails and there is a problem (Error will be in LastError)</returns>
public static DataShare CreateDataShare(string name = null, string owner = null, bool readOnly = false, bool priv = false)

/// Create a new Data Share using an existing DataPack / DataSHare / DataStore and store it in the internal list of Data Stores.
/// < "dataPack">The DataPack / DataShare / DataStore to use to populate Data Items in the DataShare</param>
/// < "name">The optional Name of the Data Share</param>
/// < "owner">The Owner of the Data Share. This is required for private Data Stores</param>
/// < "readOnly">Is this Data Share read only after having been posted?</param>
/// < "priv">Is this Data Share private (restricted to it's Owner) or not? Defaults to public</param>
/// <returns>The new DataShare just created or null if it fails and there is a problem (Error will be in LastError)</returns>
public static DataShare CreateDataShare(DataPack dataPack, string name = null, string owner = null, bool readOnly = false, bool priv = false)

General Functions

/// Close the supplied DataShare and remove it from our list in memory (if it is not found then the currently active DataShare will be closed instead)

/// < "dataShare">The DataShare to close. If it is null the currently active DataShare will be closed instead</param>
/// < "force">Force this DataShare to be removed from the list of available Headers</param>
/// <returns>True if the closure was successful otherwise false (Error will be in LastError)(</returns>
public static bool CloseDataShare(DataShare dataShare = null, bool force = false)

/// Close all the DataShares that we have opened and not yet closed
/// < "force">Force this DataShare to be removed from the list of available Headers</param>
/// <returns>True if the closures were successful otherwise false (Error will be in LastError)(</returns>
public static bool CloseAllDataShares(bool force = false)

/// Show a simple form to edit the Data Items in the supplied DataShare (which is then made the currently active DataShare). If no DataShare is supplied then the currently active DataStare is used instead
/// < "dataShare">The DataShare to edit (if no DataShare is supplied then the currently active DataStare is used instead). If the editing is cancelled out of the DataShare copy will be returned unchanged</param>
/// <returns>The edited (and now currently active) DataShare or the original unchanged if the editing was cancelled</returns>
public static DataShare EditDataShare(DataShare dataShare = null)

/// List the all the DataItems in the DataShare supplied. If one is not supplied then the currently active DataShare is shown instead
/// < "DataShare">The Data Share to list all the Data Items in. If it is null the currently active DataShare will be shown instead</param>
public static void ListDataShare(DataShare dataShare = null)

/// Get and optionally display the List of Data Shares supplied or currently available, optionally filtering them first.
/// < "allowEdit">Optionally allow editing of the Shares in the Share List</param>
/// < "showListDialog">Show all the Data Shares found in the List</param>
/// < "showFilterDialog">Optionally show the Filter dialog form to filter the Shares that are displayed</param>
/// < "showSortDialog">Optionally show the Sort dialog form to sort the order in which the Shares are displayed</param>
/// <returns>A List of all the Headers of all the Data Shares found in memory</returns>
public static List<Header> ListDataShares(bool allowEdit = false, bool showListDialog = false, bool showFilterDialog = false, bool showSortDialog = false)

/// Extract and optionally remove a DataPack DataItem from this DataShare
/// < "dataShare">The DataShare to extract from (it will default to the currently active DataShare if not supplied)</param>
/// < "name">The name of the DataPack DataItem to extract. Use the Path delimiter '/' to access an embedded / nested DataPack</param>
/// < "remove">Optionally remove the DataPack DataItem from the parent DataPack (defaults to false)</param>
/// <returns>The extracted DataPack or null if it could not be found</returns>
public static DataPack ExtractDataPack(DataShare dataShare, string name, bool remove = false)

/// Save either the supplied or the currently active DataShare to disk, optionally supplying the file name. Note that any existing file will be automatically overwritten
/// < "dataShare">The DataShare to save to disk. If it is null the currently active DataShare will be saved instead</param>
/// < "filename">An optional name for the file. If not supplied then either the last file name used (if available) will be used or a new file name will be created / determined</param>
/// < "showSaveDialog">Optionally show the Windows File Save dialog to select a path and specify a name if no valid file name is available (defaults to false)</param>
/// <returns>True if the DataShare was successfully saved to disk otherwise false (with the problem in Data.LastError)</returns>
public static bool SaveDataShare(DataShare dataShare = null, string filename = null, bool showSaveDialog = false)

/// Add Header / Meta Data items to either the supplied DataShare or the currently active DataShare
/// <"dataShare">The optional DataShare to add the Header data to. If it is null the currently active DataShare will be used</param>
/// <returns>The supplied or the currently active DataShare with the header / meta data added</returns>
public static DataShare AddDataShareHeaderData(DataShare dataShare = null)

/// List the Header / Meta Data Items in the DataShare supplied. If one is not supplied then values from the currently active DataShare is shown instead
/// <param name="dataShare">The Data Share to list all the Header / Meta Data Items in. If it is null the currently active DataShare will be used instead</param>
public static void ListDataShareHeaderData(DataShare dataShare = null)

/// Convert the DataShare supplied into a new DataStore. If it is null then the currently active Data Share will be used
/// < "dataShare">The DataShare to use to populate the DataStore (If it is null then the currently active Data Share will be used)</param>
/// < "showCreateDialog">Optionally show the create DataStore dialog form to enter additional information</param>
/// < "name">The name of the Data Store (which may be used to help define the file name) (not required if showDialog is set to true)</param>
/// < "path">The optional path / directory where you want this Data Store to be saved to / read from (not required if showDialog is set to true)</param>
/// <returns>The (currently active) DataStore just created or null if there was a problem (Error will be in LastError)</returns>
public static DataStore ConvertDataShareToDataStore(DataShare dataShare = null, bool showCreateDialog = false, string name = null, string path = null)

Post / Publish Functions

/// Post or optionally repost the DataShare, so as to make it available to other processes, either the DataShare passed or the currently active DataShare if null
/// < "dataShare">The DataShare to Post so as to make it available to other processes (If it is null then the currently active Data Share will be used)</param>
/// < "allowRepost">Allow this DataShare to be posted again, replacing the previous version if it has already been posted (defaults to false)

/// <returns>The index of the DataShare in the DataShare Header List or -1 if the Post failed for any reason</returns>
public static int PostDataShare(DataShare dataShare = null, bool allowRepost = false)

/// Repost either the DataShare passed or the currently active DataShare if null, updating the contents of a previously Posted (or Reposted) DataShare
/// Note: It will Post the DataShare if it has not already been Posted.
/// < "dataShare">The DataShare to Repost / Update so as to make it available to other processes (If it is null then the currently active Data Share will be used)</param>
/// <returns>The index of the DataShare in the DataShare Header List or -1 if the Repost failed for any reason</returns>
public static int RepostDataShare(DataShare dataShare = null)

/// Update either the DataShare passed or the currently active DataShare if null, updating the contents of a previously Posted (or Reposted) DataShare
/// Note: Update is similar to Repost except that it will fail if the DataShare has not already been Posted (unlike Repost which will simply Post it instead)
/// < "dataShare">The DataShare to Repost / Update so as to make it available to other processes (If it is null then the currently active Data Share will be used)</param>
/// <returns>True if the DataShare was successfully Updated, otherwise false</returns>
public static bool UpdateDataShare(DataShare dataShare = null)

Open Functions

/// Open the DataShare identified by its slot in the Data Share List and make it the currently active DataShare
/// <”slot">The slot the requested DataShare occupies in the Data Share List</param>
/// <”getLock">Get the DataShare Lock so that no-one else can update it until we release the lock (defaults to false)</param>
/// <returns>The DataShare requested or null if there was a problem (details in LastError)</returns>
public static DataShare OpenDataShare(int slot, bool getLock = false)

/// Open the DataShare identified by its ID and make it the currently active DataShare
/// <”id">The ID that uniquely identified the requested DataShare</param>
/// <”getLock">Get the DataShare Lock so that no-one else can update it until we release the lock (defaults to false)</param>
/// <returns>The DataShare requested or null if there was a problem (details in LastError)</returns>
public static DataShare OpenDataShare(long id, bool getLock = false)

/// Open the DataShare identified by its Name and make it the currently active DataShare
/// <”name">The Name of the requested DataShare</param>
/// <”getLock">Get the DataShare Lock so that no-one else can update it until we release the lock (defaults to false)</param>
/// <returns>The DataShare requested or null if there was a problem (details in LastError)</returns>
public static DataShare OpenDataShare(string name, bool getLock = false)

Get and Find Functions

/// Get the Data Item at the specified index in the currently active or supplied DataShare
/// <”index">The index of the Data Item required (must be greater than -1 and less than the Item Count)</param>
/// <”dataShare">The optional DataShare to get the Data Item from or null to use the currently active DataShare</param>
/// <returns>The DataItem at the specified index or null if there was a problem (a description of which will be in Data.LastError)</returns>
public static DataItem getDataShareItem(int index, DataShare dataShare= null)

/// Get the Data Item with the specified name in the currently active or supplied DataShare
/// <”name">The name of the Data Item required</param>
/// <”dataShare">The optional DataShare to get the Data Item from or null to use the currently active DataShare</param>
/// <returns>The DataItem with the specified name or null if there was a problem (a description of which will be in Data.LastError)</returns>
public static DataItem getDataShareItem(string name, DataShare dataShare = null)

/// Find the Data Item at the specified index in this DataShare
/// <”index">The index of the Data Item required (must be greater than -1 and less than the Item Count)</param>
/// <”dataShare">The optional DataShare to get the Data Item from or null to use the currently active DataShare</param>
/// <returns>The DataItem at the specified index or null if there was a problem (a description of which will be in Data.LastError)</returns>
public static DataItem findDataShareItem(int index, DataShare dataShare = null)

/// Find the Data Item with the specified name in this DataShare
/// <”name">The name of the Data Item required</param>
/// <”dataShare">The optional DataShare to get the Data Item from or null to use the currently active DataShare</param>
/// <returns>The DataItem with the specified name or null if there was a problem (a description of which will be in Data.LastError)</returns>
public static DataItem findDataShareItem(string name, DataShare dataShare = null)

DataStore Functions

Constructors

/// Show the create DataStore dialog and use it to create a new DataStore and make it the current DataStore.
/// <returns>The new DataStore just created or null if it fails or there is a problem (Error will be in LastError)</returns>
public static DataStore CreateDataStore()

/// Show the create DataStore dialog and use it to create a new DataStore using the supplied DataPack, make it the current DataStore
/// < "dataPack">The DataPack to use to populate the DataStore</param>
/// <returns>The new DataStore just created or null if it fails or there is a problem (Error will be in LastError)</returns>
public static DataStore CreateDataStore(DataPack dataPack)

/// Show the create DataStore dialog and use it to create a new DataStore using the supplied DataShare, make it the current DataStore
/// <”dataShare">The DataShare to use to populate the DataStore</param>
/// <returns>The new DataStore just created or null if it fails or there is a problem (Error will be in LastError)</returns>
public static DataStore CreateDataStore(DataShare dataShare)

/// Create a new Data Store with a name and optional path
/// <”name">The optional name of the Data Store (may be used to help define the file name)</param>
/// <”path">The optional path / directory where you want this Data Store to be saved to / read from</param>
public static DataStore CreateDataStore(string name = null, string path = null)

/// Create a new Data Store containing the supplied Data Pack Data Items with an optional name and path. If no DataPack is specified then the currently active DataPack will be used instead
/// <”dataPack">The Data Pack containing the Data Items to store. If not specified then the currently active DataPack will be used instead</param>
/// <”name">The optional name of the Data Store (may be used to help define the file name)</param>
/// <”path">The optional path / directory where you want this Data Store to be saved to / read from</param>
public static DataStore CreateDataStore(DataPack dataPack, string name = null, string path = null)

/// Create a new Data Store containing the supplied Data Share Data Items with an optional name and path. If no DataShare is specified then the currently active DataShare will be used instead
/// <”dataShare">The Data Share containing the Data Items to store. If not specified then the currently active DataShare will be used instead</param>
/// <”name">The optional name of the Data Store (may be used to help define the file name)</param>

/// <”path">The optional path / directory where you want this Data Store to be saved to / read from</param>
public static DataStore CreateDataStore(DataShare dataShare, string name = null, string path = null)

General Functions

/// Show a simple form to edit the Data Items in the supplied DataStore (which is then made the currently active DataStore). If no DataStore is specified then the currently active DataStore will be used instead
/// <”dataStore">The DataStore to edit (if not specified then the currently active DataStore will be used instead). If the editing is cancelled out of the DataStore will be returned unchanged</param>
/// <returns>The edited DataStore or the original unchanged if the editing was cancelled</returns>
public static DataStore EditDataStore(DataStore dataStore = null)

/// List the all the DataItems in the DataStore supplied. If no DataStore is specified then the currently active DataStore will be used instead
/// <”dataStore">The Data Store to list the contents of. If not specified then the currently active DataStore will be used instead</param>
public static void ListDataStore(DataStore dataStore = null)

/// Extract and optionally remove a DataPack DataItem from this DataStore
/// < "dataStore">The DataStore to extract from (it will default to the currently active DataShare if not supplied)</param>
/// < "name">The name of the DataPack DataItem to extract. Use the Path delimiter '/' to access an embedded / nested DataPack</param>
/// <"remove">Optionally remove the DataPack DataItem from the parent DataPack (defaults to false)</param>
/// <returns>The extracted DataPack or null if it could not be found</returns>
public static DataPack ExtractDataPack(DataStore dataStore, string name, bool remove = false)

/// Add Header / Meta Data items to either the supplied DataStore or the currently active DataStore
/// <”dataStore">The optional DataStore to add the Header data to. If it is null the currently active DataStore will be used</param>
/// <returns>The supplied or the currently active DataStore with the header / meta data added</returns>
public static DataStore AddDataStoreHeaderData(DataStore dataStore = null)

/// List the Header / Meta Data Items in the DataStore supplied. If one is not supplied then values from the currently active DataStore is shown instead
/// <”dataStore">The DataStore to list all the Header / Meta Data Items in. If it is null the currently active DataStore will be used instead</param>
public static void ListDataStoreHeaderData(DataStore dataStore = null)

/// Close the supplied DataStore. If this is null then the currently active DataStore will be closed instead

/// <”dataStore">The DataStore to close. If it is null the currently active DataStore will be closed instead</param>
/// <”save">Save the DataStore to disk if it has been changed (defaults to true)</param>
/// <returns>True if the closure was successful otherwise false (Error will be in LastError)(</returns>
public static bool CloseDataStore(DataStore dataStore = null, bool save = true)

/// Delete the supplied DataStore and remove it from disk (if it was saved). If this is null then the currently active DataStore will be deleted instead
/// <”dataStore">The DataStore to delete and remove from disk. If it is null the currently active DataStore will be deleted instead</param>
/// <returns>True if the deletion was successful otherwise false (Error will be in LastError)(</returns>
public static bool DeleteDataStore(DataStore dataStore = null)

/// Convert the DataStore supplied into a new DataShare. If it is null then the currently active Data Store will be used
/// <”dataStore">The DataStore to use to populate the DataShare (If it is null then the currently active Data Store will be used)</param>
/// <”showCreateDialog">Optionally show the create DataShare dialog form to enter additional information</param>
/// <”name">The optional Name of the Data Share (not required if showCreateDialog is set to true)</param>
/// <”owner">The Owner of the Data Share. This is required for private Data Stores (not required if showCreateDialog is set to true)</param>
/// <”readOnly">Is this Data Share read only after having been posted? (not required if showCreateDialog is set to true)</param>
/// <”priv">Is this Data Share private (restricted to it's Owner) or not? Defaults to public (not required if showCreateDialog is set to true)</param>
/// <returns>The (currently active) DataShare just created or null if there was a problem (Error will be in LastError)</returns>
public static DataShare ConvertDataStoreToDataShare(DataStore dataStore = null, bool showCreateDialog = false, string name = null, string owner = null, bool readOnly = false, bool priv = false)

Open and Save Functions

/// Open a new Data Store from a file on disk using a valid name and path and make it the currently active DataStore. Optionally showing the Windows File Open Dialog
/// Note that showOpenDialog will automatically be set to true if no name and path is specified. If no path is specified then the user's default path will be used.
/// Note also that if the currently active DataStore has changes that need saving then that will happen automatically first (if possible).
/// <”name">The name of the Data Store to open (used to help define the file name)</param>
/// <”path">The optional path / directory where you want this Data Store to be read from (will default to the user's home directory if not specified)</param>
/// <”showOpenDialog">Optionally show the Windows File Open Dialog to select the file to open (defaults to false)</param>
/// <returns>The newly opened DataStore if successful or null if there was a problem (detailed in LastError)</returns>

public static DataStore OpenDataStore(string name = null, string path = null, bool showOpenDialog = false)

/// Open a new Data Store as a DataPack from a file on disk using a valid name and path and make it the currently active DataPack. Optionally showing the Windows File Open Dialog
/// Note that showOpenDialog will automatically be set to true if no name and path is specified. If no path is specified then the user's default path will be used.
/// <”name">The name of the Data Store to open (used to help define the file name)</param>
/// <”path">The optional path / directory where you want this Data Store to be read from (will default to the user's home directory if not specified)</param>
/// <”showOpenDialog">Optionally show the Windows File Open Dialog to select the file to open (defaults to false)</param>
/// <returns>The newly opened DataPack if successful or null if there was a problem (detailed in LastError)</returns>
public static DataPack OpenAsDataPack(string name = null, string path = null, bool showOpenDialog = false)

/// Open a new Data Store as a DataShare from a file on disk using a valid name and path and make it the currently active DataShare. Optionally showing the Windows File Open Dialog
/// <”name">The name of the Data Store to open (used to help define the file name)</param>
/// <”path">The optional path / directory where you want this Data Store to be read from (will default to the user's home directory if not specified)</param>
/// <”showOpenDialog">Optionally show the Windows File Open Dialog to select the file to open (defaults to false)</param>
/// <”showCreateDialog">Setting this will also open the create DataShare dialog to add additional information</param>
/// <returns>The newly opened DataShare if successful or null if there was a problem (detailed in LastError)</returns>
public static DataShare OpenAsDataShare(string name = null, string path = null, bool showOpenDialog = false, bool showCreateDialog = false)

/// Save either the supplied or the currently active DataStore to disk, optionally supplying the file name. Note that any existing file will be automatically overwritten
/// <”dataStore">The DataStore to save to disk. If it is null the currently active DataStore will be saved instead</param>
/// <”filename">An optional name for the file. If not supplied then either the last file name used (if available) will be used or a new file name will be created / determined</param>
/// <”showSaveDialog">Optionally show the Windows File Save dialog to select a path and specify a name if no valid file name is available (defaults to false)</param>
/// <returns>True if the DataStore was successfully saved to disk otherwise false (with the problem in Data.LastError)</returns>
public static bool SaveDataStore(DataStore dataStore = null, string filename = null, bool showSaveDialog = false)

Get and Find Functions

/// Get the Data Item at the specified index in the currently active or supplied DataStore
/// <”index">The index of the Data Item required (must be greater than -1 and less than the Item Count)</param>

/// <”dataStore">The optional DataStore to get the Data Item from or null to use the currently active DataStore</param>
/// <returns>The DataItem at the specified index or null if there was a problem (a description of which will be in Data.LastError)</returns>
public static DataItem getDataStoreItem(int index, DataStore dataStore = null)

/// Get the Data Item with the specified name in the currently active or supplied DataStore
/// <”name">The name of the Data Item required</param>
/// <”dataStore">The optional DataStore to get the Data Item from or null to use the currently active DataStore</param>
/// <returns>The DataItem with the specified name or null if there was a problem (a description of which will be in Data.LastError)</returns>
public static DataItem getDataStoreItem(string name, DataStore dataStore = null)

/// Find the Data Item at the specified index in this DataStore
/// <”index">The index of the Data Item required (must be greater than -1 and less than the Item Count)</param>
/// <”dataStore">The optional DataStore to get the Data Item from or null to use the currently active DataStore</param>
/// <returns>The DataItem at the specified index or null if there was a problem (a description of which will be in Data.LastError)</returns>
public static DataItem findDataStoreItem(int index, DataStore dataStore = null)

/// Find the Data Item with the specified name in this DataStore
/// <”name">The name of the Data Item required</param>
/// <”dataStore">The optional DataStore to get the Data Item from or null to use the currently active DataStore</param>
/// <returns>The DataItem with the specified name or null if there was a problem (a description of which will be in Data.LastError)</returns>
public static DataItem findDataStoreItem(string name, DataStore dataStore = null)

Forms Display Functions

/// Show the create DataShare dialog and use it to create a new DataShare, make it the current DataShare and store it in the internal list of Data Shares.
/// <”dataPack">The DataPack to use to populate the DataShare. If it is null the current DataPack will be used and if that is null then a new DataPack is created</param>
/// <returns>The new DataShare just created or null if it fails or there is a problem (Error will be in LastError)</returns>
public static DataShare ShowCreateDataShareForm(DataPack dataPack = null)

/// Show the create DataStore dialog and use it to create a new DataStore, make it the current DataStore and store it in the internal list of Data Stores.
/// < "dataPack">The DataPack to use to populate the DataStore. If it is null the current DataPack will be used and if that is null then a new DataPack is created</param>
/// <returns>The new DataStore just created or null if it fails or there is a problem (Error will be in LastError)</returns>
public static DataStore ShowCreateDataStoreForm(DataPack dataPack = null)

/// Show the Edit Data Pack form to edit the DataPack supplied. If the DataPack is null the current DataPack will be used and if that is null then a new DataPack will be created
/// <”dataPack">The DataPack to edit. If it is null the current DataPack will be used and if that is null then a new DataPack created</param>
/// <returns>The edited DataPack</returns>
public static DataPack ShowEditDataItemsForm(DataPack dataPack = null)

/// List the all the DataItems in the DataPack / DataShare / DataStore supplied. If one is not supplied then the currently active DataPack is shown instead
/// </summary>
/// <“DataPack">The Data Pack to list all the Data Items in. If it is null the currently active DataShare will be shown instead</param>
public static void ShowListDataItemsForm(DataPack dataPack = null)

/// Show the Add Data Item(s) Form to add items to the supplied Data Pack. If the DataPack is null the current DataPack will be used and if that is null then a new DataPack will be created
/// <”dataPack">The DataPack to add Data Items to. If it is null the current DataPack will be used and if that is null then a new DataPack created</param>
/// <returns>The edited DataPack with the new Data Items</returns>
public static DataPack ShowAddDataItemsForm(DataPack dataPack = null)

/// Show the Insert Data Item(s) Form to insert new Data Items into the supplied Data Pack
/// < "index">The Index into the DataPack to insert the Data Items</param>
/// < "dataPack">The DataPack to insert Data Items into. If it is null the current DataPack will be used and if that is null then a new DataPack will be created</param>
/// <returns>The edited DataPack with the new Data Items inserted</returns>
public static DataPack ShowInsertDataItemsForm(int index, DataPack dataPack = null)

/// Show the Edit Data Items Form to edit the specified Data Items in the supplied Data Pack
/// < "index">The Index into the DataPack of the Data Item to edit</param>
/// < "dataPack">The DataPack to edit the Data Items in. If it is null the current DataPack will be used and if that is null then null will be returned</param>

/// <returns>The edited DataPack with the new Data Items inserted</returns>
public static DataPack ShowEditDataItemForm(int index, DataPack dataPack = null)

/// Merge the supplied DataPack / DataShare / DataStore with another DataPack / DataShare / DataStore or a new DataPack or a DataStore off disk opened as a DataPack
/// < "dataPack">The DataPack / DataShare / DataStore to merge the Data Items with another DataPack / DataShare / DataStore. If it is null the current DataPack will be used and if that is null then null will be returned</param>
/// <returns>The merged DataPack with the new Data Items merged into it or the originally supplied DataPack if not merged or null if there was a problem</returns>
public static DataPack ShowMergeDataItemsForm(DataPack dataPack)

/// Display the List of Data Shares supplied or currently available, optionally filtering them first.
/// < "shares">Optionally supply the List of Data Shares to display. If null it will create its own list to display</param>
/// < "allowEdit">Allow editing of the Shares in the List</param>
/// < "showFilterDialog">Optionally show the Filter dialog form to filter the Shares that are displayed</param>
/// < "showSortDialog">Optionally show the Sort dialog form to sort the order in which the Sahes are displayed</param>
/// <returns>The number of Data Shares that were in the list</returns>
public static int ShowListDataSharesForm(List<Header> shares = null, bool allowEdit = false, bool showFilterDialog = false, bool showSortDialog = false)

/// Show the dialog box enabling setting the Filters to be applied to the DataShares List
/// < "filters">The optional starting set of filters to use</param>
/// <returns>The Filters to be applied to the List of Shares or null if there are none</returns>
public static ShareList.Filters ShowSetDataShareFilters(ShareList.Filters filters = null)

/// Show the dialog box enabling the sorting of DataShares in the Share List
/// < "sort">The optional starting sort to use</param>
/// <returns>The Sort to be applied to the List of Shares or null if there are none</returns>
public static ShareList.Sort ShowSortDataShareList(ShareList.Sort sort = null)

/// Show the List all Error Messages in history form
public static void ShowListErrorMessagesForm()

/// Show the Add Header Item(s) Form to add header items to the supplied DataShare. If the DataShare is null the current DataShare will be used and if that is null then a new DataShare will be created
/// <”dataShare">The DataShare to add the Header Data Items to. If it is null the current DataShare will be used and if that is null then a new DataShare created</param>
/// <returns>The edited DataShare with the new Data Header Items in it if items were added</returns>
public static DataShare ShowAddDataHeaderItemsForm(DataShare dataShare = null)

Full API

The full normal API consists of using the functions and properties exposed in the various classes implemented in the library directly. It involves creating and using instances of those classes. It can be used alongside the Data Class Static API (and sometimes needs to be) but care should be taken when doing so.

DataType Class

There are a number of different types of data supported. Each DataItem in a DataPack is of a supported DataType.

The type itself is capitalised such as Bool. The type definition is an enum in all caps such as BOOL:

/// The Type of Variable we are dealing with
public enum TYPE : byte

Each Data Type has the following properties available:

/// What Type of data this is
public abstract TYPE Type

/// The size of this Data Type (in bytes)
public abstract int Size

/// The number of items in the DataPack or array
public virtual int Count

There are a couple of static functions available:

/// Validate that the value supplied in a string is a valid value for the specified DataType
/// <”type">The Type of Data to validate</param>
/// <”value">The value to validate</param>
/// <returns>True if the supplied string contains a valid value of the specified type</returns>
public static bool isValidValue(TYPE type, string value)

/// Validate that the value supplied in an object is a valid value for the specified DataType
/// < "type">The Type of Data to validate</param>
/// < "value">The object value to validate</param>
/// <returns>True if the supplied object contains a valid value of the specified type</returns>
public static bool isValidValue(TYPE type, object value)

/// Get the Data Type based on the advertised name / description of that DataType (ie: Bool.Type == "Bool" or Byte.Type == "Byte")
/// <”type">The description of the DataType as a string</param>
/// <returns>The DataType that fits the description (defaults to String)</returns>
public static TYPE getDataType(string type)

The DataType class itself is abstract and so there are separate DataType classes derived from it for each supported data type:

Name	C# Type	Valid Values
Bool	bool	true / false
Byte	unsigned 8 bit integer	0 - 255
Char	char (16 bit)	U+0000 to U+FFFF
Short	short (signed 16 bit integer)	-32,768 to 32,767
Int	int (signed 32 bit integer)	-2,147,483,648 to 2,147,483,647
Long	long (signed 64-bit integer)	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Float	float (4 byte numeric)	~6-9 digits
Double	double (8 byte numeric)	~15-17 digits
Decimal	decimal (16 byte numeric)	~28-29 digits
String	string	null, blank or any string value
Datapack	DataPack	Unlimited nested DataPacks

Each DataType class has the following properties:

/// What Data Type this is (TYPE enum value)
public override TYPE Type

/// The size of the variable (in bytes) or the length of the string if it is a String Data Type
public override int Size

/// The type of the variable as a string (i.e. Bool or String)
public static string Desc

DataItem Class

Every data item stored in a DataPack / DataShare or DataStore will be an instance of the DataItem class regardless of what type of data it contains.

Constructors

Each constructor requires a value of the appropriate type and an optional name.

/// Create a new Boolean Type Data Item
/// <”value">The boolean value to set (defaults to false if not supplied)</param>
/// <”name">The (optional) name of the variable</param>
public DataItem(bool value, string name = null, string code = null)

/// Create a new Byte Type Data Item
/// <”value">The byte value to set (defaults to 0 if not supplied)</param>
/// <”name">The (optional) name of the variable</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
public DataItem(byte value, string name = null, string code = null)

/// Create a new Char Type Data Item
/// <”value">The char value to set (defaults to ' ' if not supplied)</param>
/// <”name">The (optional) name of the variable</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
public DataItem(char value, string name = null, string code = null)

/// Create a new Short Type Data Item
/// < "value">The short value to set (defaults to 0 if not supplied)</param>
/// < "name">The (optional) name of the variable</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
public DataItem(short value, string name = null, string code = null)

/// Create a new Int Type Data Item
/// < "value">The integer value to set (defaults to 0 if not supplied)</param>
/// < "name">The (optional) name of the variable</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
public DataItem(int value, string name = null, string code = null)

/// Create a new Long Type Data Item
/// < "value">The long value to set (defaults to 0 if not supplied)</param>
/// < "name">The (optional) name of the variable</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
public DataItem(long value, string name = null, string code = null)

/// Create a new Float Type Data Item
/// < "value">The float value to set (defaults to 0.0 if not supplied)</param>
/// < "name">The (optional) name of the variable</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
public DataItem(float value, string name = null, string code = null)

/// Create a new Double Type Data Item
/// < "value">The double value to set (defaults to 0.0 if not supplied)</param>
/// < "name">The (optional) name of the variable</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
public DataItem(double value, string name = null, string code = null)

/// Create a new Decimal Type Data Item
/// < "value">The decimal value to set (defaults to 0.0 if not supplied)</param>
/// < "name">The (optional) name of the variable</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
public DataItem(decimal value, string name = null, string code = null)

/// Create a new String Type Data Item
/// < "value">The string value to set (defaults to empty string if not supplied)</param>
/// < "name">The (optional) name of the variable</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
public DataItem(string value, string name = null, string code = null)

/// Create a new Datapack Type Data Item
/// < "value">The DataPack value to set (defaults to null if not supplied)</param>
/// < "name">The (optional) name of the variable</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
public DataItem(DataPack value, string name = null, string code = null)

Create Data Items

/// Static function used to create a new DataItem based on the specified type and using the value in the string input (which will be validated)
/// <”type">The Type of the DataItem to be created</param>
/// <”value">The value to put into the DataItem in a string which will be validated</param>
/// <”name">The (optional) name of the DataItem (defaults to null if not supplied)</param>
/// < "index">The index of this Data Item in the DataPack (defaults to -1 if not set)</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
<returns>The DataItem just created or null if there was a problem (error in Data.LastError)</returns>
public static DataItem CreateDataItem(DataType.TYPE type, string value, string name = null, int index = -1, string code = null)

/// Static function used to create a new DataItem based on the specified type and using the value in the supplied object (which will be validated)
/// < "type">The Type of the DataItem to be created</param>
/// < "value">The value to put into the DataItem as an object which will be validated</param>
/// < "name">The (optional) name of the DataItem (defaults to null if not supplied)</param>
/// < "index">The index of this Data Item in the DataPack (defaults to -1 if not set)</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
<returns>The DataItem just created or null if there was a problem (error in Data.LastError)</returns>
public static DataItem CreateDataItem(DataType.TYPE type, object value, string name = null, int index = -1, string code = null)

/// Create and return a new Bool Data Item from a string value
/// <"value">The string containing the value to assign to the Bool Data Item</param>
/// <"name">The (optional) name of the variable (defaults to false if not supplied or invalid)</param>
/// < "index">The index of this Data Item in the DataPack (defaults to -1 if not set)</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or the default value (false) if there was a problem</returns>
public static DataItem CreateBool(string value, string name = null, int index = -1, string code = null)

/// Create and return a new Byte Data Item from a string value
/// < "value">The string containing the value to assign to the Byte Data Item</param>
/// < "name">The (optional) name of the variable (defaults to 0 if not supplied or invalid)</param>
/// < "index">The index of this Data Item in the DataPack (defaults to -1 if not set)</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or the default value (0) if there was a problem</returns>
public static DataItem CreateByte(string value, string name = null, int index = -1, string code = null)

/// Create and return a new Char Data Item from a string value
/// < "value">The string containing the value to assign to the Short Data Item</param>
/// < "name">The (optional) name of the variable (defaults to ' ' if not supplied or invalid)</param>
/// < "index">The index of this Data Item in the DataPack (defaults to -1 if not set)</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or the default value (' ')if there was a problem</returns>
public static DataItem CreateChar(string value, string name = null, int index = -1, string code = null)

/// Create and return a new Short Data Item from a string value
/// < "value">The string containing the value to assign to the Short Data Item</param>
/// <"name">The (optional) name of the variable (defaults to 0 if not supplied or invalid)</param>

/// < "index">The index of this Data Item in the DataPack (defaults to -1 if not set)</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or the default value (0) if there was a problem</returns>
public static DataItem CreateShort(string value, string name = null, int index = -1, string code = null)

/// Create and return a new Int Data Item from a string value
/// < "value">The string containing the value to assign to the Int Data Item</param>
/// < "name">The (optional) name of the variable (defaults to 0 if not supplied or invalid)</param>
/// < "index">The index of this Data Item in the DataPack (defaults to -1 if not set)</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or the default value (0) if there was a problem</returns>
public static DataItem CreateInt(string value, string name = null, int index = -1, string code = null)

/// Create and return a new Long Data Item from a string value
/// < "value">The string containing the value to assign to the Long Data Item</param>
/// < "name">The (optional) name of the variable (defaults to 0 if not supplied or invalid)</param>
/// < "index">The index of this Data Item in the DataPack (defaults to -1 if not set)</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or the default value (0) if there was a problem</returns>
public static DataItem CreateLong(string value, string name = null, int index = -1, string code = null)

/// Create and return a new Float Data Item from a string value
/// < "value">The string containing the value to assign to the Float Data Item</param>
/// < "name">The (optional) name of the variable (defaults to 0 if not supplied or invalid)</param>
/// < "index">The index of this Data Item in the DataPack (defaults to -1 if not set)</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or the default value (0) if there was a problem</returns>
public static DataItem CreateFloat(string value, string name = null, int index = -1, string code = null)

/// Create and return a new Double Data Item from a string value
/// < "value">The string containing the value to assign to the Double Data Item</param>
/// < "name">The (optional) name of the variable (defaults to 0 if not supplied or invalid)</param>
/// < "index">The index of this Data Item in the DataPack (defaults to -1 if not set)</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or the default value (0) if there was a problem</returns>
public static DataItem CreateDouble(string value, string name = null, int index = -1, string code = null)

/// Create and return a new Decimal Data Item from a string value
/// < "value">The string containing the value to assign to the Decimal Data Item</param>
/// < "name">The (optional) name of the variable (defaults to 0 if not supplied or invalid)</param>
/// < "index">The index of this Data Item in the DataPack (defaults to -1 if not set)</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or the default value (0) if there was a problem</returns>
public static DataItem CreateDecimal(string value, string name = null, int index = -1, string code = null)

/// Create and return a new String Data Item
/// < "value">The string value to assign to the String Data Item</param>
/// < "name">The (optional) name of the variable (defaults to null if not supplied)</param>
/// < "index">The index of this Data Item in the DataPack (defaults to -1 if not set)</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or null if there was a problem</returns>
public static DataItem CreateString(string value, string name = null, int index = -1, string code = null)

/// Create and return a new Datapack Data Item from an object value
/// < "value">The DataPack object containing the value to assign to the Datapack Data Item</param>
/// < "name">The (optional) name of the variable (defaults to 0 if not supplied or invalid)</param>
/// < "index">The index of this Data Item in the DataPack (defaults to -1 if not set)</param>
/// < "code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The DataItem just created or the default value (null) if there was a problem</returns>
public static DataItem CreateDatapack(DataPack value, string name = null, int index = -1, string code = null)

Properties

/// The Type of variable being stored in this Data Item
public DataType.TYPE Type

/// The Type of variable being stored in this Data Item as a byte value
public byte TypeValue

/// The Size / Length of the variable
public int Size

/// The Name of the Variable (if it has one)
public string Name

/// An optional field containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.

public string Code

/// The Boolean Value of this Data Item
public bool BoolValue

/// The Byte Value of this Data Item
public byte ByteValue

/// The Char Value of this Data Item
public char CharValue

/// The Short Value of this Data Item
public short ShortValue

/// The Int Value of this Data Item
public int IntValue

/// The Long Value of this Data Item
public long LongValue

/// The Float Value of this Data Item
public float FloatValue

/// The Double Value of this Data Item
public double DoubleValue

/// The Decimal Value of this Data Item
public decimal DecimalValue

/// The String Value of this Data Item
public string StringValue

/// The DataPack value of this Data Item
public DataPack DatapackValue

/// The actual value of the Data Item as an Object
public object Value

/// The value of the Data Item as a String
public string AsStringValue

/// Get a description of the Type of the Data Item
public string Desc

DataPack Class

The DataPack class is used to contain, maintain and manipulate Data Items of any type. There is no limit on the number that may be stored in a DataPack. The DataShare and DataStore classes are derived from it and so support all its functionality.

Note that this class is now derived from IEnumerable which will allow it to be used in foreach loops and the like.

API Changes: *When I first created DataPacks they were just a convenient way of packaging disparate types of data (they still are). However, over time as I have expanded their functionality and used them more extensively their capabilities have become more and more like that of .NET Collection classes such as List, ArrayList etc. Unfortunately its API is anything but like the normal .NET Collection Classes API.*

As a result I made the decision to change the DataPack API to become more like the industry standard (and simpler). For example addDataItems becomes AddRange however the function parameters remain the same. The old legacy API is still there but not generally compiled into the MVShareStore library, a separate version containing both the old legacy API and the new API is available. Both are detailed in this document with the original legacy API calls first and the new replacement function names second.

Constructors

/// Create a new empty DataPack with the specified capacity
/// <“capacity">The (optional) expected capacity of the DataPack</param>
public DataPack(int capacity = 0)

/// Create a new DataPack and populate it using the supplied DataPack (using deep copy to create a duplicate).
/// < "dataPack">The DataPack to duplicate in this DataPack</param>
public DataPack(DataPack dataPack)

/// Create a new DataPack and populate it using the supplied array of DataItems.
/// < "dataItems">The array of DataItems to use to populate the DataPack</param>
public DataPack(DataItem [] dataItems)

/// Create a new DataPack and populate it using the supplied List of DataItems.
/// <"dataItems">The List of DataItems to use to populate the DataPack</param>
public DataPack(List<DataItem> dataItems)

Add Data Items

/// Add another DataItem to the list of data items stored in the DataPack and return it's index in the list
/// < "dataItem">The DataItem to add to the DataPack</param>
/// < "ignoreDups">Ignore duplicate Item names (normally only used for Config files</param>
/// <returns>The index of the item in the DataPack (or -1 if there was a problem)</returns>

public int addDataItem(DataItem dataItem, bool ignoreDups = false)
public int Add(DataItem dataItem, bool ignoreDups = false)

/// Add another DataItem to the list of data items stored in the DataPack and return it's index in the list
/// <”type">The Type of the DataItem to be added</param>
/// <”value">The value to put into the DataItem in a string which will be validated</param>
/// <”name">The (optional) name of the DataItem (defaults to null if not supplied)</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The index of the item in the DataPack (or -1 if there was a problem)</returns>
public int addDataItem(DataType.TYPE type, string value, string name = null , string code = null)
public int Add(DataType.TYPE type, string value, string name = null , string code = null)

/// Add another DataItem to the list of data items stored in the DataPack and return it's index in the list
/// < "type">The Type of the DataItem to be added</param>
/// < "value">The value to put into the DataItem in an object which will be validated</param>
/// < "name">The (optional) name of the DataItem (defaults to null if not supplied)</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The index of the item in the DataPack (or -1 if there was a problem)</returns>
public int addDataItem(DataType.TYPE type, object value, string name = null , string code = null)
public int Add(DataType.TYPE type, object value, string name = null , string code = null)

/// Add a Bool DataItem to the list of data items stored in the DataPack and return it's index in the list
/// <”value">The boolean value to put into the DataItem</param>
/// <”name">The (optional) name of the DataItem (defaults to null if not supplied)</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The index of the item in the DataPack (or -1 if there was a problem)</returns>
public int addDataItem(bool value, string name = null , string code = null)
public int Add(bool value, string name = null , string code = null)

/// Add a Byte DataItem to the list of data items stored in the DataPack and return it's index in the list
/// <”value">The byte value to put into the DataItem</param>
/// <”name">The (optional) name of the DataItem (defaults to null if not supplied)</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The index of the item in the DataPack (or -1 if there was a problem)</returns>
public int addDataItem(byte value, string name = null , string code = null)
public int Add(byte value, string name = null , string code = null)

/// Add a Char DataItem to the list of data items stored in the DataPack and return it's index in the list
/// <“value">The char value to put into the DataItem</param>
/// <“name">The (optional) name of the DataItem (defaults to null if not supplied)</param>

/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The index of the item in the DataPack (or -1 if there was a problem)</returns>
public int addDataItem(char value, string name = null , string code = null)
public int Add(char value, string name = null , string code = null)

/// Add a Short DataItem to the list of data items stored in the DataPack and return it's index in the list
/// <“value">The short value to put into the DataItem</param>
/// <“name">The (optional) name of the DataItem (defaults to null if not supplied)</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The index of the item in the DataPack (or -1 if there was a problem)</returns>
public int addDataItem(short value, string name = null , string code = null)
public int Add(short value, string name = null , string code = null)

/// Add an Int DataItem to the list of data items stored in the DataPack and return it's index in the list
/// <“value">The integer value to put into the DataItem</param>
/// <“name">The (optional) name of the DataItem (defaults to null if not supplied)</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The index of the item in the DataPack (or -1 if there was a problem)</returns>
public int addDataItem(int value, string name = null , string code = null)
public int Add(int value, string name = null , string code = null)

/// Add a Long DataItem to the list of data items stored in the DataPack and return it's index in the list
/// <“value">The long value to put into the DataItem</param>
/// <“name">The (optional) name of the DataItem (defaults to null if not supplied)</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The index of the item in the DataPack (or -1 if there was a problem)</returns>
public int addDataItem(long value, string name = null , string code = null)
public int Add(long value, string name = null , string code = null)

/// Add a Float DataItem to the list of data items stored in the DataPack and return it's index in the list
/// <“value">The float value to put into the DataItem</param>
/// <“name">The (optional) name of the DataItem (defaults to null if not supplied)</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The index of the item in the DataPack (or -1 if there was a problem)</returns>
public int addDataItem(float value, string name = null , string code = null)
public int Add(float value, string name = null , string code = null)

/// Add a Double DataItem to the list of data items stored in the DataPack and return it's index in the list
/// <“value">The double value to put into the DataItem</param>
/// <“name">The (optional) name of the DataItem (defaults to null if not supplied)</param>

/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The index of the item in the DataPack (or -1 if there was a problem)</returns>
public int addDataItem(double value, string name = null , string code = null)
public int Add(double value, string name = null , string code = null)

/// Add a Decimal DataItem to the list of data items stored in the DataPack and return it's index in the list
/// <“value">The decimal value to put into the DataItem</param>
/// <“name">The (optional) name of the DataItem (defaults to null if not supplied)</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The index of the item in the DataPack (or -1 if there was a problem)</returns>
public int addDataItem(decimal value, string name = null , string code = null)
public int Add(decimal value, string name = null , string code = null)

/// Add a String DataItem to the list of data items stored in the DataPack and return it's index in the list
/// <“value">The string value to put into the DataItem</param>
/// <“name">The (optional) name of the DataItem (defaults to null if not supplied)</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The index of the item in the DataPack (or -1 if there was a problem)</returns>
public int addDataItem(string value, string name = null , string code = null)
public int Add(string value, string name = null , string code = null)

/// Add a DataPack DataItem to the list of data items stored in the DataPack and return it's index in the list
/// < "value">The DataPack to put into the DataItem</param>
/// < "name">The (optional) name of the DataItem (defaults to null if not supplied)</param>
/// < "count">The current or expected Item Count for the Datapack DataItem</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>The index of the item in the DataPack (or -1 if there was a problem)</returns>
public int addDataItem(DataPack value, string name = null, int count = 0 , string code = null)
public int Add(DataPack value, string name = null, int count = 0 , string code = null)

/// Add additional DataItems from another DataPack to the list of data items stored in this DataPack and return the number added
/// <“dataPack">The DataPack to add the Items from to this DataPack</param>
/// <returns>The number of new Items added</returns>
public int addDataItems(DataPack dataPack)
public int AddRange(DataPack dataPack)

/// Add additional DataItems from a List of Data Items to the Data Items already stored in this DataPack and return the number added
/// <“dataItems">The List of Data Items to add to the DataPack</param>
/// <returns>The number of new Items added</returns>
public int addDataItems(List<DataItem> dataItems)
public int AddRange(List<DataItem> dataItems)

/// Add additional DataItems from an array of Data Items to the list of Data Items already stored in this DataPack and return the number added
/// <“dataItems">The array of Data Items to add to the DataPack</param>
/// <returns>The number of new Items added</returns>
public int addDataItems(DataItem [] dataItems)
public int AddRange(DataItem [] dataItems)

Add or Set Data Items

/// Set the Data Item with the specified name to the value supplied. If not found then add the Data Item instead.
/// <"name">The name of the Data Item to change or add</param>
/// <"value">The Data Item to set the value to</param>
/// <returns>True if successful otherwise false</returns>
public bool addOrSetDataItem(string name, DataItem value)
public bool AddOrSet (string name, DataItem value)

/// Set the Bool Data Item with the specified name to the value supplied. If not found then add the Data Item instead
/// <"name">The name of the Data Item to change or add</param>
/// <"value">The boolean value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool addOrSetDataItem(string name, bool value , string code = null)
public bool AddOrSet (string name, bool value , string code = null)

/// Set the Byte Data Item with the specified name to the value supplied. If not found then add the Data Item instead
/// <"name">The name of the Data Item to change or add</param>
/// <"value">The byte value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool addOrSetDataItem(string name, byte value, string code = null)
public bool AddOrSet(string name, byte value, string code = null)

/// Set the Char Data Item with the specified name to the value supplied. If not found then add the Data Item instead
/// <"name">The name of the Data Item to change or add</param>
/// <"value">The char value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool addOrSetDataItem(string name, char value, string code = null)
public bool AddOrSet (string name, char value, string code = null)

/// Set the Short Data Item with the specified name to the value supplied. If not found then add the Data Item instead

/// <"name">The name of the Data Item to change or add</param>
/// <"value">The short value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool addOrSetDataItem(string name, short value, string code = null)
public bool AddOrSet (string name, short value, string code = null)

/// Set the Integer Data Item with the specified name to the value supplied. If not found then add the Data Item instead
/// <”name">The name of the Data Item to change or add</param>
/// <”value">The integer value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool addOrSetDataItem(string name, int value, string code = null)
public bool AddOrSet (string name, int value, string code = null)

/// Set the Long Data Item with the specified name to the value supplied. If not found then add the Data Item instead
/// <”name">The name of the Data Item to change or add</param>
/// <”value">The long value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool addOrSetDataItem(string name, long value, string code = null)
public bool AddOrSet (string name, long value, string code = null)

/// Set the Float Data Item with the specified name to the value supplied. If not found then add the Data Item instead
/// <”name">The name of the Data Item to change or add</param>
/// <”value">The float value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool addOrSetDataItem(string name, float value, string code = null)
public bool AddOrSet (string name, float value, string code = null)

/// Set the Double Data Item with the specified name to the value supplied. If not found then add the Data Item instead
/// <”name">The name of the Data Item to change or add</param>
/// <”value">The double value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool addOrSetDataItem(string name, double value, string code = null)
public bool AddOrSet (string name, double value, string code = null)

/// Set the Decimal Data Item with the specified name to the value supplied. If not found then add the Data Item instead
/// <”name">The name of the Data Item to change or add</param>

/// <”value">The decimal value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool addOrSetDataItem(string name, decimal value, string code = null)
public bool AddOrSet (string name, decimal value, string code = null)

/// Set the String Data Item with the specified name to the value supplied. If not found then add the Data Item instead
/// <”name">The name of the Data Item to change or add</param>
/// <”value">The string value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool addOrSetDataItem(string name, string value, string code = null)
public bool AddOrSet(string name, string value, string code = null)

/// Set the Datapack Data Item with the specified name to the value supplied. If not found then add the Data Item instead
/// <”name">The name of the Data Item to change or add</param>
/// <”value">The DataPack value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool addOrSetDataItem(string name, DataPack value, string code = null)
public bool AddOrSet (string name, DataPack value, string code = null)

Set Data Items

/// Set the Data Item at the specified index to the value supplied
/// <“index">The index of the Data Item to change</param>
/// <“value">The new Data Item to set the Data Item to</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(int index, DataItem value)
public bool Set(int index, DataItem value)

/// Set the Data Item with the specified name to the value supplied
/// <“name">The name of the Data Item to change</param>
/// <“value">The new Data Item to set the Data Item to</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(string name, DataItem value)
public bool Set(string name, DataItem value)

/// Set the Bool Data Item at the specified index to the value supplied
/// <“index">The index of the Data Item to change</param>
/// <“value">The boolean value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(int index, bool value, string code = null)

public bool Set(int index, bool value, string code = null)

/// Set the Bool Data Item with the specified name to the value supplied
/// <“name">The name of the Data Item to change</param>
/// <“value">The boolean value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(string name, bool value, string code = null)
public bool Set(string name, bool value, string code = null)

/// Set the Byte Data Item at the specified index to the value supplied
/// <“index">The index of the Data Item to change</param>
/// <“value">The byte value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(int index, byte value, string code = null)
public bool Set(int index, byte value, string code = null)

/// Set the Byte Data Item with the specified name to the value supplied
/// <“name">The name of the Data Item to change</param>
/// <“value">The byte value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(string name, byte value, string code = null)
public bool Set(string name, byte value, string code = null)

/// Set the Char Data Item at the specified index to the value supplied
/// <“index">The index of the Data Item to change</param>
/// <“value">The char value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(int index, char value, string code = null)
public bool Set(int index, char value, string code = null)

/// Set the Char Data Item with the specified name to the value supplied
/// <“name">The name of the Data Item to change</param>
/// <“value">The char value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(string name, char value, string code = null)
public bool Set(string name, char value, string code = null)

/// Set the Short Data Item at the specified index to the value supplied
/// <“index">The index of the Data Item to change</param>
/// <“value">The short value to set the Data Item to</param>

/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(int index, short value, string code = null)
public bool Set(int index, short value, string code = null)

/// Set the Short Data Item with the specified name to the value supplied
/// <“name">The name of the Data Item to change</param>
/// <“value">The short value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(string name, short value, string code = null)
public bool Set(string name, short value, string code = null)

/// Set the Int Data Item at the specified index to the value supplied
/// <“index">The index of the Data Item to change</param>
/// <“value">The integer value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(int index, int value, string code = null)
public bool Set(int index, int value, string code = null)

/// Set the Int Data Item with the specified name to the value supplied
/// <“name">The name of the Data Item to change</param>
/// <“value">The integer value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(string name, int value, string code = null)
public bool Set(string name, int value, string code = null)

/// Set the Long Data Item at the specified index to the value supplied
/// <“index">The index of the Data Item to change</param>
/// <“value">The long value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(int index, long value, string code = null)
public bool Set(int index, long value, string code = null)

/// Set the Long Data Item with the specified name to the value supplied
/// <“name">The name of the Data Item to change</param>
/// <“value">The long value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(string name, long value, string code = null)
public bool Set(string name, long value, string code = null)

/// Set the Float Data Item at the specified index to the value supplied
/// <“index">The index of the Data Item to change</param>
/// <“value">The float value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(int index, float value, string code = null)
public bool Set(int index, float value, string code = null)

/// Set the Float Data Item with the specified name to the value supplied
/// <“name">The name of the Data Item to change</param>
/// <“value">The float value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(string name, float value, string code = null)
public bool Set(string name, float value, string code = null)

/// Set the Double Data Item at the specified index to the value supplied
/// <“index">The index of the Data Item to change</param>
/// <“value">The double value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(int index, double value, string code = null)
public bool Set(int index, double value, string code = null)

/// Set the Double Data Item with the specified name to the value supplied
/// <“name">The name of the Data Item to change</param>
/// <“value">The double value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(string name, double value, string code = null)
public bool Set(string name, double value, string code = null)

/// Set the Decimal Data Item at the specified index to the value supplied
/// <“index">The index of the Data Item to change</param>
/// <“value">The decimal value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(int index, decimal value, string code = null)
public bool Set(int index, decimal value, string code = null)

/// Set the Decimal Data Item with the specified name to the value supplied
/// <“name">The name of the Data Item to change</param>
/// <“value">The decimal value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>

public bool setDataItem(string name, decimal value, string code = null)
public bool Set(string name, decimal value, string code = null)

/// Set the String Data Item at the specified index to the value supplied
/// <“index">The index of the Data Item to change</param>
/// <“value">The string value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(int index, string value, string code = null)
public bool Set(int index, string value, string code = null)

/// Set the String Data Item with the specified name to the value supplied
/// <“name">The name of the Data Item to change</param>
/// <“value">The string value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(string name, string value, string code = null)
public bool Set(string name, string value, string code = null)

/// Set the Datapack Data Item at the specified index to the value supplied
/// < "index">The index of the Data Item to change</param>
/// < "value">The DataPack value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(int index, DataPack value, string code = null)
public bool Set(int index, DataPack value, string code = null)

/// Set the Datapack Data Item with the specified name to the value supplied
/// < "name">The name of the Data Item to change</param>
/// < "value">The DataPack value to set the Data Item to</param>
/// <”code">An optional value containing whatever is desired. i.e. A maint field, a section or subset name, an update flag etc.</param>
/// <returns>True if successful otherwise false</returns>
public bool setDataItem(string name, DataPack value, string code = null)
public bool Set(string name, DataPack value, string code = null)

/// Set the value of the Code field for the Data Item found at the specified index.
/// <"index">The index of the Data Item to change the Code field in</param>
/// <"code">The new value of the Code field</param>
/// <returns>The Data Item with the modified Code value or null if it was not found</returns>
public DataItem SetCode(int index, string code)

/// Set the value of the Code field for the Data Item found with the specified name.
/// <"name">The name of the Data Item to change the Code field in</param>
/// <”code">The new value of the Code field</param>
/// <returns>The Data Item with the modified Code value or null if it was not found</returns>
public DataItem SetCode(string name, string code)

Find and Get Functions

/// Get the Data Item at the specified index in this DataPack
/// <“index">The index of the Data Item required (must be greater than -1 and less than the Item Count)</param>
/// <returns>The DataItem at the specified index or null if there was a problem (a description of which will be in Data.LastError)</returns>
public DataItem getDataItem(int index)
public DataItem GetItem(int index)

/// Get the Data Item with the specified name in this DataPack
/// <“name">The name of the Data Item required</param>
/// <returns>The DataItem with the specified name or null if there was a problem (a description of which will be in Data.LastError)</returns>
public DataItem getDataItem(string name)
public DataItem GetItem(string name)

/// Get the boolean value from the Data Item at the index specified
/// <”index">The index of the Data Item's value required (must be greater than -1 and less than the Item Count)</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The boolean value of the DataItem at the specified index or the default value (false) if there was a problem</returns>
public bool getBoolValue(int index, bool defValue = false)
public bool GetBool (int index, bool defValue = false)

/// Get the boolean value from the Data Item identified by the specified name
/// <”name">The name of the Data Item containing the value desired</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The boolean value of the DataItem with the specified name or the default value (false) if there was a problem</returns>
public bool getBoolValue(string name, bool defValue = false)
public bool GetBool (string name, bool defValue = false)

/// Get the byte value from the Data Item at the index specified
/// <”index">The index of the Data Item's value required (must be greater than -1 and less than the Item Count)</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The byte value of the DataItem at the specified index or the default value (0) if there was a problem</returns>
public byte getByteValue(int index, byte defValue = 0)
public byte GetByte (int index, byte defValue = 0)

/// Get the byte value from the Data Item identified by the specified name
/// <”name">The name of the Data Item containing the byte value desired</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The byte value of the DataItem with the specified name or the default value (0) if there was a problem</returns>
public byte getByteValue(string name, byte defValue = 0)
public byte GetByte(string name, byte defValue = 0)

/// Get the char value from the Data Item at the index specified
/// <”Index">The index of the Data Item's value required (must be greater than -1 and less than the Item Count)</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The char value of the DataItem at the specified index or the default value (' ') if there was a problem</returns>
public char getCharValue(int index, char defValue = ' ')
public char GetChar(int index, char defValue = ' ')

/// Get the char value from the Data Item identified by the specified name
/// <”name">The name of the Data Item containing the char value desired</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The char value of the DataItem with the specified name or the default value (0) if there was a problem</returns>
public char getCharValue(string name, char defValue = ' ')
public char GetChar(string name, char defValue = ' ')

/// Get the short value from the Data Item at the index specified
/// <”index">The index of the Data Item's value required (must be greater than -1 and less than the Item Count)</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The short value of the DataItem at the specified index or the default value (0) if there was a problem</returns>
public short getShortValue(int index, short defValue = 0)
public short GetShort(int index, short defValue = 0)

/// Get the short value from the Data Item identified by the specified name
/// <”name">The name of the Data Item containing the byte value desired</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The short value of the DataItem with the specified name or the default value (0) if there was a problem</returns>
public short getShortValue(string name, short defValue = 0)
public short GetShort(string name, short defValue = 0)

/// Get the integer value from the Data Item at the index specified
/// <”index">The index of the Data Item's value required (must be greater than -1 and less than the Item Count)</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The integer value of the DataItem at the specified index or the default value (0) if there was a problem</returns>
public int getIntValue(int index, int defValue = 0)
public int GetInt(int index, int defValue = 0)

/// Get the integer value from the Data Item identified by the specified name
/// <”name">The name of the Data Item containing the integer value desired</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The integer value of the DataItem with the specified name or the default value (0) if there was a problem</returns>
public int getIntValue(string name, int defValue = 0)
public int GetInt(string name, int defValue = 0)

/// Get the long value from the Data Item at the index specified
/// <”index">The index of the Data Item's value required (must be greater than -1 and less than the Item Count)</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The long value of the DataItem at the specified index or the default value (0) if there was a problem</returns>
public long getLongValue(int index, long defValue = 0)
public long GetLong(int index, long defValue = 0)

/// Get the long value from the Data Item identified by the specified name
/// <”name">The name of the Data Item containing the integer value desired</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The long value of the DataItem with the specified name or the default value (0) if there was a problem</returns>
public long getLongValue(string name, long defValue = 0)
public long GetLong(string name, long defValue = 0)

/// Get the float value from the Data Item at the index specified
/// <”index">The index of the Data Item's value required (must be greater than -1 and less than the Item Count)</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The float value of the DataItem at the specified index or the default value (0) if there was a problem</returns>
public float getFloatValue(int index, float defValue = 0)
public float GetFloat(int index, float defValue = 0)

/// Get the float value from the Data Item identified by the specified name
/// <”name">The name of the Data Item containing the integer value desired</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The float value of the DataItem with the specified name or the default value (0) if there was a problem</returns>
public float getFloatValue(string name, float defValue = 0)
public float GetFloat(string name, float defValue = 0)

/// Get the double value from the Data Item at the index specified
/// <”index">The index of the Data Item's value required (must be greater than -1 and less than the Item Count)</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The double value of the DataItem at the specified index or the default value (0) if there was a problem</returns>
public double getDoubleValue(int index, double defValue = 0)
public double GetDouble(int index, double defValue = 0)

/// Get the double value from the Data Item identified by the specified name
/// <”name">The name of the Data Item containing the integer value desired</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The double value of the DataItem with the specified name or the default value (0) if there was a problem</returns>
public double getDoubleValue(string name, double defValue = 0)
public double GetDouble (string name, double defValue = 0)

/// Get the decimal value from the Data Item at the index specified
/// <”index">The index of the Data Item's value required (must be greater than -1 and less than the Item Count)</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The decimal value of the DataItem at the specified index or the default value (0) if there was a problem</returns>
public decimal getDecimalValue(int index, decimal defValue = 0)
public decimal GetDecimal(int index, decimal defValue = 0)

/// Get the decimal value from the Data Item identified by the specified name
/// <”name">The name of the Data Item containing the integer value desired</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The decimal value of the DataItem with the specified name or the default value (0) if there was a problem</returns>
public decimal getDecimalValue(string name, decimal defValue = 0)
public decimal GetDecimal(string name, decimal defValue = 0)

/// Get the string value from the Data Item at the index specified
/// <”index">The index of the Data Item's value required (must be greater than -1 and less than the Item Count)</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The string value of the DataItem at the specified index or the default value (empty string) if there was a problem</returns>
public string getStringValue(int index, string defValue = "")
public string GetString(int index, string defValue = "")

/// Get the string value from the Data Item identified by the specified name
/// <”name">The name of the Data Item containing the string value desired</param>
/// <”defValue">The default value to use if the DataItem is not found</param>
/// <returns>The string value of the DataItem with the specified name or the default value (empty string) if there was a problem</returns>
public string getStringValue(string name, string defValue = "")
public string GetString(string name, string defValue = "")

/// Get the DataPack value from the Data Item at the index specified
/// <”index">The index of the Data Item's value required (must be greater than -1 and less than the Item Count)</param>
/// <”defValue">The default value to use if the DataItem is not found (if left null then an empty DataPack will be returned)</param>
/// <returns>The DataPack value of the DataItem with the specified name or the default value (empty DataPack) if there was a problem</returns>
public DataPack getDatapackValue(int index, DataPack defValue = null)
public DataPack GetDatapack(int index, DataPack defValue = null)

/// Get the DataPack value from the Data Item identified by the specified name
/// <”name">The name of the Data Item containing the integer value desired</param>
/// <”defValue">The default value to use if the DataItem is not found (if left null then an empty DataPack will be returned)</param>
/// <returns>The DataPack value of the DataItem with the specified name or the default value (empty DataPack) if there was a problem</returns>
public DataPack getDatapackValue(string name, DataPack defValue = null)

public DataPack GetDatapack(string name, DataPack defValue = null)

/// Get the value as an object from the Data Item at the index specified
/// <”index">The index of the Data Item's value required (must be greater than -1 and less than the Item Count)</param>
/// <”defValue">The default value to use if the DataItem is not found (if left null then new object will be returned)</param>
/// <returns>The object value of the DataItem with the specified name or the default value (new object) if there was a problem</returns>
public object getValue(int index, object defValue = null)
public object GetValue(int index, object defValue = null)

/// Get the object value from the Data Item identified by the specified name
/// <”name">The name of the Data Item containing the integer value desired</param>
/// <”defValue">The default value to use if the DataItem is not found (if left null then new object will be returned)</param>
/// <returns>The object value of the DataItem with the specified name or the default value (new object) if there was a problem</returns>
public object getValue(string name, object defValue = null)
public object GetValue(string name, object defValue = null)

/// Get the first Data Item in the DataPack (index = 0)
/// <returns>The first DataItem in the DataPack or null if there was a problem</returns>
public DataItem getFirst()
public DataItem GetFirst()

/// Is there another DataItem in the DataPack?
/// <returns>True if getNext will return the next DataItem in the DataPack</returns>
public bool hasNext()
public bool HasNext()

/// Get the next Data Item in the DataPack
/// <returns>The next DataItem in the DataPack or null if there was a problem</returns>
public DataItem getNext()
public DataItem GetNext()

/// Find the Data Item at the specified index in this DataPack
/// <“index">The index of the Data Item required (must be greater than -1 and less than the Item Count)</param>
/// <returns>The DataItem at the specified index or null if there was a problem (a description of which will be in Data.LastError)</returns>
public DataItem findDataItem(int index)
public DataItem Find(int index)

/// Find the Data Item with the specified name in this DataPack
/// <“name">The name of the Data Item required</param>
/// <returns>The DataItem with the specified name or null if there was a problem (a description of which will be in Data.LastError)</returns>
public DataItem Find(string name)

/// Find and return the specified DataPack, optionally searching a path for a nested DataPack

/// <”name">The Name of the DataPack Data Item to search for</param>
/// <”path">The path to a nested DataPack (path separator is a / if required). Defaults to null</param>
/// <returns>The specified DataPack or null if it was not found</returns>
public DataPack findDataPack(string name, string path = null)
public DataPack FindDataPack(string name, string path = null)

Insert Data Items

/// Insert the supplied DataItem into the DataPack at the specified index / location
/// <“index">The index in the DataPack to insert this DataItem (if an item at that index does not currently exist then it will be added instead)</param>
/// <“dataItem">The DataItem to insert into the DataPack</param>
/// <returns>The indexof the inserted DataItem or -1 if there was a problem</returns>
public int insertDataItem(int index, DataItem dataItem)
public int Insert(int index, DataItem dataItem)

/// Insert additional DataItems from another DataPack into the list of data items stored in this DataPack at the specified index and return the number added
/// <“index">The index in the DataPack to start the insertion of Data Items from (if an item at that index does not currently exist then it will be added instead)</param>
/// <“dataPack">The DataPack to insert the Items from into this DataPack</param>
/// <returns>The number of new Items added</returns>
public int insertDataItems(int index, DataPack dataPack)
public int InsertRange(int index, DataPack dataPack)

/// Insert additional DataItems from List of DataItems into the list of data items stored in this DataPack at the specified index and return the number added
/// <“index">The index in the DataPack to start the insertion of Data Items from (if an item at that index does not currently exist then it will be added instead)</param>
/// <“dataItems">The LIst of DataItems to insert the Items from into this DataPack</param>
/// <returns>The number of new Items added</returns>
public int insertDataItems(int index, List<DataItem> dataItems)
public int InsertRange(int index, List<DataItem> dataItems)

/// Insert additional DataItems from an array of DataItems into the list of data items stored in this DataPack at the specified index and return the number added
/// <“index">The index in the DataPack to start the insertion of Data Items from (if an item at that index does not currently exist then it will be added instead)</param>
/// <“dataItems">The LIst of DataItems to insert the Items from into this DataPack</param>
/// <returns>The number of new Items added</returns>
public int insertDataItems(int index, DataItem [] dataItems)
public int InsertRange(int index, DataItem [] dataItems)

Edit Data Items

/// Edit the Data Item at the specified index into the DataPack replacing it with the supplied Data Item
/// <“index">The index in the DataPack of the Data Item to change</param>

/// <“dataItem">The Data Item to replace the existing Data Item with</param>
/// <returns>The replaced Data Item or null if there was a problem</returns>
public DataItem editDataItem(int index, DataItem dataItem)
public DataItem Edit(int index, DataItem dataItem)

/// Edit the Data Item at the specified index into the DataPack replacing it with the supplied Data Item
/// <“name">The original name of the Data Item to change (to enable it to be found)</param>
/// <“dataItem">The Data Item to replace the existing Data Item with</param>
/// <returns>The replaced Data Item or null if there was a problem</returns>
public DataItem editDataItem(string name, DataItem dataItem)
public DataItem Edit(string name, DataItem dataItem)

/// Delete the Data Item at the specified index into the DataPack
/// <”index">The index in the DataPack of the Data Item to delete</param>
/// <returns>The number of Data Items remaining or -1 if there was a problem</returns>
public int deleteDataItem(int index)
public int RemoveAt(int index)

/// Delete the Data Item at the specified index into the DataPack
/// <”name">The name of the Data Item to delete</param>
/// <returns>The number of Data Items remaining or -1 if there was a problem</returns>
public int deleteDataItem(string name)
public int Remove(string name)

/// Empty / Clear all the Data Items from this DataPack
public void clearDataItems()
public void Clear()

Merge, Extract and Sort Functions

/// Merge the Data Items of 2 DataPacks together and return the resulting DataPack
/// <“dataPack1">The DataPack to have the second DataPack merged into</param>
/// <“dataPack2">The DataPack to merge into the first DataPack</param>
/// <returns>The resulting merged DataPack or null if there was a problem</returns>
public static DataPack MergeDataItems(DataPack dataPack1, DataPack dataPack2)
public static DataPack Merge (DataPack dataPack1, DataPack dataPack2)

/// Merge the DataPack passed into this DataPack.
/// <“dataPack">The DataPack to merge with this DataPack</param>
/// <returns>The number of new Data Items added by the merge</returns>
public int mergeDataItems(DataPack dataPack)
public int Merge(DataPack dataPack)

/// Extract and optionally remove a DataPack DataItem from this DataPack
/// <”name">The name of the DataPack DataItem to extract. Use the Path delimiter '/' to access an embedded / nested DataPack</param>
/// <”remove">Optionally remove the DataPack DataItem from the parent DataPack (defaults to false)</param>
/// <returns>The extracted DataPack or null if it could not be found</returns>

public DataPack extractDataPack(string name, bool remove = false)
public DataPack Extract(string name, bool remove = false)

/// Sort the DataPack passed and return it in sorted ascending or descending Data Item Name order (unnamed Data Items will all be at the beginning for ascending sorts and at the end for descending sorts)
/// <“dataPack">The DataPack to sort and return</param>
/// <“descending">Should the sort order be descending (defaults to ascending)</param>
/// <returns>The sorted DataPack or null if there was a problem</returns>
public static DataPack SortDataItems(DataPack dataPack, bool descending = false)
public static DataPack Sort(DataPack dataPack, bool descending = false)

/// Sort the DataPack into ascending or descending Data Item Name order (unnamed Data Items will all be at the beginning for ascending sorts and at the end for descending sorts)
/// <“descending">Should the sort order be descending (defaults to ascending)</param>
/// <returns>The number of DataItems sorted if it was successful otherwise -1 if there was a problem</returns>
public int sortDataItems(bool descending = false)
public int Sort(bool descending = false)

General Functions

/// Close this DataPack and clear out any Data Items
public void Close()

/// Get the Index of the Data Item with the specified name
/// <“name">The name of the Data Item to search for</param>
/// <returns>The index of the specified Data Item or -1 if it was not found or there was a problem</returns>
public int getItemIndex(string name)
public int IndexOf(string name)

/// Does a Data Item using this name already exist in the DataPack
/// <“name">The name to check in the list</param>
/// <returns>True if a Data Item using this name already exists in the Data Pack</returns>
public bool nameExists(string name)
public bool Exists(string name)

/// Does the DataPack contain an entry with this name?
/// <”name">The name to check in the list</param>
/// <returns>True if a Data Item using this name already exists in the Data Pack</returns>
public bool Contains(string name)

Properties

/// What is the derived class type? DataPack, DataShare or DataStore.
public enum TYPE
    DATAPACK
    DATASHARE

DATASTORE

/// What type of DataPack is this? DataPack, DataShare or DataStore?
public TYPE Type

/// Get the number of Data Items currently held in the DataPack
public int Count

/// Get the number of Data Items currently held in the DataPack
public int ItemCount

/// Get the total number of Data Items in this DataPack plus any nested DataPacks.
public int TotalCount

/// Has this DataPack been changed since it was first created / saved?
public bool Dirty

/// The number of Data Items in the DataPack
public int Size

/// The total size of all the Data Items in the DataPack added together including nested datapacks
public int TotalSize

/// The current (last added, found or set) Data Item
public DataItem CurrentItem

/// The Data Items contained in this DataPack
public List<DataItem> Items

/// All the Data Items contained in this DataPack, including Data Items in nested DataPacks unpacked.
/// Note that an embedded DataPack with be returned as an empty Data Item with the number of Data Items in that DataPack in the Count field
/// This will be followed by all the Data Items contained in that nested DataPack.
/// Data Item Index numbers will be assigned and will be in the order that the Data Items are read and returned.
public List<DataItem> AllItems

/// Enumerate through all the Data Items in this DataPack using a foreach loop.
/// <returns>Every Data Item in this DataPack</returns>
public IEnumerator GetEnumerator()

/// Iterate through all the Data Items this DataPack using a foreach loop.
/// <returns>Every DataItem Item in this DataPack</returns>
public IEnumerable GetItems

/// Iterate through all the Data Items and nested DataItems in this DataPack using a foreach loop.
/// <returns>Every DataItem Item (including nested) in this DataPack</returns>
public IEnumerable AllDataItems

public IEnumerable GetAllItems

/// Do any of the Data Items in this DataPack / DataShare / DataStore contain values in the Code field?
public bool HasCodes

/// A description of the Code values in this DataPack (will be used for column headers etc.)
public string CodeDescription

/// Print a summary of the Item Count, Total Count (including nested DataPacks) and the Total Size of the Data Items (also including nested DataPacks)
/// <returns>The Data Item Count, Total Count and the Total Size of the Data Items in a string</returns>
public override string ToString()

/// Return all the DataItems in the DataPack as an array
/// <”incNested">Also include all the Data Items nested in the DataPack's Data Items (defaults to false)</param>
/// <returns>All the Data Items in the DataPack as an array</returns>
public DataItem[] ToArray(bool incNested = false)

/// The capacity of the current list of Data Items
public int Capacity

DataShare Class

The DataShare class is derived from the DataPack class and therefore inherits all its functionality. It implements additional capabilities enabling it to be used to post / publish DataPacks so that they can be viewed and / or manipulated by other processes on the same host. Each posted DataShare will appear in the Share List on the host.

DataShares can have an optional name, optional owner / group / user and be marked as private to that owner / group / user and can also be marked as read only.

Constructors

/// Create a (deep) copy of the supplied DataShare
/// <“dataShare">The DataShare to make a copy of</param>
public DataShare(DataShare dataShare)

/// Create a new Data Share using optional parameters
/// <“name">The optional Name of the Data Share</param>
/// <“readOnly">Is this Data Share read only after having been posted? Defaults to false</param>
/// <“private">Is this Data Share private (restricted to it's Owner) or not? Defaults to public</param>
/// <“owner">The Owner of the Data Share. This is required for private Data Stores</param>
public DataShare(string name = null, string owner = null, bool readOnly = false, bool priv = false)

/// Create a new Data Share of the specified type. If no type is specified it will default to Indexed
/// <“dataPack">The DataPack to use to populate this DataShare</param>
/// <“name">The optional Name of the Data Share</param>
/// <“owner">The Owner of the Data Share. This is required for private Data Stores</param>
/// <“readOnly">Is this Data Share read only after having been posted?</param>
/// <“priv">Is this Data Share private (restricted to it's Owner) or not? Defaults to public</param>
public DataShare(DataPack dataPack, string name = null, string owner = null, bool readOnly = false, bool priv = false)

General Functions

/// Close this Data Share
/// <“force">Force this DataShare to be removed from the list of available Headers</param>
/// <“remove">Internal Use Only</param>
/// <returns>True if this Data Share was successfully closed otherwise false (with the reason in Data.LastError)</returns>
public bool Close(bool force = false, bool remove = true)

/// Close all the DataShares that we have opened and not yet closed
/// <“force">Force this DataShare to be removed from the list of available Headers</param>

/// <returns>True if they were all successfully closed</returns>
public static bool CloseAll(bool force = false)

/// Set the Name of the DataShare to the supplied value
/// <“name">The new Name for the DataShare (or null to remove it)</param>
/// <returns>The new Name of the DataShare</returns>
public string setName(string name)

/// Set the Owner of the DataShare to the supplied value
/// <“name">The new Owner of the DataShare (or null to remove it)</param>
/// <returns>The new Owner of the DataShare</returns>
public string setOwner(string owner)

/// Post or optionally repost this DataShare so that it can be shared with other processes
/// <“allowRepost">Allow this DataShare to be posted again, replacing the previous version if it has already been posted (defaults to false)</param>
/// <returns>The Slot in the Share List that it has been assigned or -1 if there was a problem</returns>
public int Post(bool allowRepost = false)

/// Repost this DataShare, updating the contents of a previously Posted DataShare so that it can be shared with other processes
/// Note: It will Post the DataShare instead if it has not already been Posted.
/// <“waitForLock">Should this continue waiting to get the lock forever (defaults to false). If false it will get the lock after the timeout</param>
/// <returns>The Slot in the Share List that it has been assigned or -1 if there was a problem</returns>
public int Repost(bool waitForLock = false)

/// Update this DataShare, updating the contents of a previously Posted DataShare so that it can be shared with other processes
/// Note: It will fail if the DataShare has not already been Posted.
/// <“waitForLock">Should this continue waiting to get the lock forever (defaults to false). If false it will get the lock after the timeout</param>
/// <returns>True if the DataShare was successfully Updated, otherwise false</returns>
public bool Update(bool waitForLock = false)

/// Increment the DataShare User Count
/// <returns>The DataShare User Count after incrementation</returns>
public int incUserCount()

/// Decrement the DataShare User Count
/// <returns>The DataShare User Count after decrementation</returns>
public int decUserCount()

/// Lock this DataShare ready for editing (assuming it has been Posted that is)
/// <“waitForever">Should this continue waiting to get the lock forever (defaults to false)</param>
/// <returns>True if the lock was successfully got and set</returns>
public bool getLockForEditing(bool waitForever = false)

/// Release the Header Lock for this DataShare (assuming it has the lock that is)
/// <returns>The current value of this DataShare's Lock</returns>
public void releaseLock()

Static Open Functions

/// Open the DataShare identified by the slot in the Header Share List
/// <“slot">The slot in the Header Share List that the Header resides in</param>
/// <“getLock">Get the DataShare Lock so that no-one else can update it until we release the lock (defaults to false)</param>
/// <returns>The DataShare just opened or null if there was a problem</returns>
public static DataShare Open(int slot, bool getLock = false)

/// Open the DataShare identified by the supplied ID
/// <“id">The ID of the DataShare to open</param>
/// <“getLock">Get the DataShare Lock so that no-one else can update it until we release the lock (defaults to false)</param>
/// <returns>The DataShare just opened or null if there was a problem</returns>
public static DataShare Open(long id, bool getLock = false)

/// Open the DataShare identified by the supplied Name
/// <“name">The Name of the DataShare to open</param>
/// <“getLock">Get the DataShare Lock so that no-one else can update it until we release the lock (defaults to false)</param>
/// <returns>The DataShare just opened or null if there was a problem</returns>
public static DataShare Open(string name, bool getLock = false)

Properties

/// The Name used to identify this Data Share
/// Note: Duplicate DataStore names in the same ShareStore are allowed but not recommended as a search by name will only ever return the first DataShare found with that name.
/// </summary>
public string Name

/// The maximum allowed length of DataShare Names
public static int MaxNameLength

/// The Owner of the Data Share (Server, Client or User). Required if it is marked Private
public string Owner

/// The maximum allowed length of DataShare Names
public static int MaxOwnerLength

/// The numeric ID uniquely identifying this DataShare.
public long ID

/// Is this Data Share read only after having been posted?
public bool ReadOnly

/// Is this Data Share Private and require a password to read and write to?
public bool Private

/// Has the DataShare been posted into global memory for sharing?
public bool Posted

/// The DataShare Header containing information regarding this DataShare.
public Header Header

/// The (optional) DataHeader that can be incorporated into every DataShare and is invisibly transferred over with it every time the DataShare is opened.
public DataHeader DataHeader

/// Get the number of Data Items currently held in the DataShare (reducing the number by one if there is also a Data Header that is currently listed in a ShareList)
public new int ItemCount

Header Class

Each DataShare created will have an instance of its associated Header class in it. This class is responsible for providing much of the information about the current state of the DataShare. It is also what is put into the ShareList when a DataShare is posted / published.

/// Create a new DataShare Header of the specified type
/// <“parent">The DataShare to which this Header belongs</param>
public Header(DataShare parent)

/// The numeric ID uniquely identifying this DataShare.
public long ID

/// The size of each Header instance (in bytes)
public static int HeaderSize

/// The number of Data Items / Variables in the Data Share
public int ItemCount

/// The number of processes (Servers & Clients) currently using this Data Share
public int Users

/// The Date / Time the Data Share was created / posted / last updated.
public long DateStamp

/// Have the contents of this DataShare been modified / updated or not?
/// Note: If the DataShare has not been Posted then this has little effect and will not update the isDirty property in the DataShare itself, only the Dirty flag in the Header.
/// If the DataShare has been Posted then this will return and / or update the 'Online' DataShare Dirty flag that is visible to other processes.
public bool Dirty

/// Is this Data Share currently locked by a process
public bool Locked

/// Is this DataShare read only so it cannot be updated after it has been posted?
public bool ReadOnly

/// Is this Data Share Private and require a password to read and write to?
public bool Private

/// The Name used to identify this Data Share.
/// Note: Duplicate DataStore names in the same ShareStore are allowed but not recommended as a search by name will only ever return the first DataShare found with that name.
public string Name

/// The Owner of the Data Share (Server, Client or User). Required if it is marked Private
public string Owner

/// Has the DataShare this Header belongs to been Posted?
public bool Posted

ShareList Class

Each DataShare that is Posted / Published is placed in the Share Store's ShareList class. The DataShares listed in this class can be accessed using static functions. How these DataShares are listed / displayed can be filtered and / or sorted using the Filter and Sort sub classes.

/// The capacity of the Data Share List (how many Headers it can contain before running out of room)
public static int Capacity

/// Get the number of Data Share Headers in the Share List
public static int Count

/// All the Data Share Headers in the List
public static List<Header> Shares

/// Get all the DataShare Headers in the Global Header List
/// <“filters">Optionally set filters regarding what is returned in the Data Share Header List</param>
/// <“sortBy">Optionally specify the field you want to sort the Share List by and whether it should be in ascending or descending order</param>
/// <returns>All the Headers in the Global Header List optionally filtered and / or sorted or an empty list if there weren't any or null if there was a problem</returns>
public static List<Header> GetShareList(Filters filters = null, Sort sortBy = null)

/// Filter the supplied Share List according to the filters that have been set and supplied
/// <“filters">The Filters to apply to the Share List</param>
/// <“shares">The list of Shares to be filtered</param>
/// <returns>The amended / filtered list of Shares</returns>
public static List<Header> FilterShareList(Filters filters, List<Header> shares = null)

/// Sort the List of Data Shares, optionally getting a new List if one has not been supplied
/// <“sortBy">The Sort instance that defines what field to sort by</param>
/// <“shares">The optional Share List to sort (a new list will be retrieved if one is not supplied)</param>
/// <returns>Either the Data Share List sorted by the field defined in the Sort instance or a new List similarly sorted</returns>
public static List<Header> SortShareList(Sort sortBy, List<Header> shares = null)

Filters Sub Class

/// Create a new Filters Class to define which Shares should be returned in a list of Shares
/// <“publicOnly">Optionally only include Shares not marked as Private</param>
/// <“privateOnly">Optionally only include Shares marked as Private</param>
/// <“writeOnly">Optionally only include Shares not marked as Read Only</param>
/// <“readOnly">Optionally only include Shares marked as Read Only</param>
/// <“dirtyOnly">Optionally only include Shares not marked as having been modified since they were posted</param>
/// <“unlockedOnly">Optionally only include Shares that are not currently locked by a process</param>
/// <“ownerOnly">Optionally only include Shares belonging to this Owner / Group</param>
/// <“excludeOwner">Optionally exclude all Shares belonging to this Owner / Group</param>
public Filters(bool publicOnly = false, bool privateOnly = false, bool writeOnly = false, bool readOnly = false, bool dirtyOnly = false, bool unlockedOnly = false, string ownerOnly = null, string excludeOwner = null)

/// Filter the Share List according to the currently set Filters and return the filtered List
/// <”shares">The List of Shares to filter</param>
/// <returns>The amended Share List after it has been filtered</returns>
public List<Header> filterList(List<Header> shares)

/// Only list and return the Shares not marked as Private
public bool PublicOnly

/// Only List and return the Shares marked as Private
public bool PrivateOnly

/// Only List and return the Shares marked as Read / Write and not Read Only
public bool WriteOnly

/// Only List and return the Shares marked as Read Only
public bool ReadOnly

/// Only List and return the Shares that have been modified since they were posted
public bool DirtyOnly

/// Only List and return the Shares marked as not currently being Locked by a process
public bool UnlockedOnly

/// Only return the Shares belonging to this particular Owner / Group
public string OwnerOnly

/// Exclude any Shares with this Owner / Group from the list returned
public string ExcludeOwner

Sort Sub Class

/// What Share Header field do we want to sort the Share List by?
public enum SORTBY
    ID
    SLOT
    NAME
    OWNER
    DATESTAMP

/// Create a new Share List Sort instance specifying how you want the Data Shares sorted
/// <”sortBy">The enum specifying the Share header field to sort by</param>
/// <”descending">Whether the sorting should be by descending order (ascending is the default)</param>
public Sort(SORTBY sortBy, bool descending = false)

/// Sort the supplied list of Shares into the order requested and return the sorted List
/// <“shares">The Share List of headers to sort</param>
/// <returns>THe sorted list of Shares or null if there was a problem</returns>
public List<Header> sortList(List<Header> shares)

/// The currently selected field to Sort by
public SORTBY SortBy

DataStore Class

The DataStore class is derived from the DataPack class and so inherits all its functionality. This class adds the capability of saving the Data Items it contains to a file on disk and being able to read them back later.

Constructors

/// Create a new Data Store with an optional name and path
/// <“name">The optional name of the Data Store (may be used to help define the file name)</param>
/// <“path">The optional path / directory where you want this Data Store to be saved to / read from</param>
public DataStore(string name = null, string path = null)

/// Create a new Data Store containing the supplied Data Pack Data Items with an optional name and path
/// <“dataPack">The Data Pack containing the Data Items to store</param>
/// <“name">The optional name of the Data Store (may be used to help define the file name)</param>
/// <“path">The optional path / directory where you want this Data Store to be saved to / read from</param>
public DataStore(DataPack dataPack, string name = null, string path = null)

/// Create a new Data Store containing the supplied Data Share Data Items with an optional name and path
/// <“dataShare">The Data Share containing the Data Items to store</param>
/// <“name">The optional name of the Data Store (may be used to help define the file name). If not specified the DataShare name will be used instead</param>
/// <“path">The optional path / directory where you want this Data Store to be saved to / read from</param>
public DataStore(DataShare dataShare, string name = null, string path = null)

General Functions

/// Close this Data Store (but leave any saved files on disk)
/// <“save">Save the DataStore to a file on disk first if it needs it (defaults to true)</param>
/// <“showDialog">Optionally show the Windows File Save dialog to select a path and specify a file name if no valid file name is available (defaults to false)</param>
/// <returns>True if this Data Store was successfully closed otherwise false (with the reason in Data.LastError)</returns>
public bool Close(bool save = true, bool showDialog = false)

/// Delete this Data Store from disk
/// <returns>True if this Data Store was successfully deleted otherwise false (with the reason in Data.LastError)</returns>
public bool Delete()

/// Open the specified DataStore, optionally supplying the file name. If no filename is specified then the File Open dialog will be displayed
/// <”filename">An optional name for the file (including the path). If not supplied then the Open File Dialog will be presented for a file to be selected</param>
/// <returns>The opened DataStore if the DataStore was successfully opened and its Data Items loaded otherwise null (with the problem in Data.LastError)</returns>
public DataStore Open(string filename = null)

/// Save this DataStore to disk, optionally supplying the file name. Note that any existing file will be automatically overwritten
/// <“filename">An optional name for the file (including the path). If not supplied then either the last file name used (if available) will be used or a new file name will be created / determined</param>
/// <“showDialog">Optionally show the Windows File Save dialog to select a path and specify a name if no valid file name is available (defaults to false)</param>
/// <returns>True if the DataStore was successfully saved to disk otherwise false (with the problem in Data.LastError)</returns>
public bool Save(string filename = null, bool showDialog = false)

/// Get the name of the Config file to open (not available in the original MVConfigFile API)
/// <”defpath">The default path / directory to use to search for Config files</param>
/// <”defname">The (optional) default file name to use. If it includes path information do not specify any value in defpath</param>
/// <returns>The name of the file selected, the name of the last file opened or null if there was a problem</returns>
public static string SelectFile(string defpath = null, string defname = null)

Properties

/// The name of the Data Store (may be used to help define the file name)
public string Name

/// The path / directory of this Data Store which is where it will be saved to / read from
public string Path

/// The full name of the file on disk where this Data Store will be saved to / read from
public string FileName

/// Was this originally a Config File before it was converted to a DataStore?
public bool wasXMLConfigFile

/// The (optional) DataHeader that can be incorporated into every DataStore and is invisibly transferred over with it every time the DataStore is opened.
public DataHeader DataHeader

DataHeader Class

The DataHeader class encapsulates the Header DataPack record that can optionally be specified and added to the DataHeader property in DataShares and DataStores as hidden / meta data that is either shared or stored with the normal Data Items but not displayed and not considered part of the Data.

Any number or type of Data Items (including DataPacks) can be included with the hidden Header / Meta Data and transferred with the normal Data.

Constructors

/// Create a new DataHeader record, optionally specifying the DataPack to use to populate it.
/// <”dataPack">The optional DataPack to use to populate the Data Header</param>
public DataHeader(DataPack dataPack = null)

/// Create a new DataHeader record, optionally specifying the DataPack to use to populate it.
/// <param name="parent">The parent DataShare that owns this DataHeader</param>
/// <param name="dataPack">The optional DataPack to use to populate the Data Header</param>
public DataHeader(DataShare parent, DataPack dataPack = null)

Properties

/// The DataPack that contains all the Header Data.
/// Adding records to this DataPack is the only way to create Header / Meta Data records and view them afterwards.
public DataPack Data

ShareStore Class

A ShareStore is where all the posted / published DataShares reside and is created automatically (using the default name) if required. However multiple ShareStores are allowed on the same host as long as they have different names.

General Functions

/// Create a new or open an existing Share Store optionally using the specified name and maximum lengths for the DataStore Name and Owner fields
/// <“name">The (optional name for the ShareStore</param>
/// <“maxNameLen">The maximum length that DataShare Names can be. If 0 or not specified will default to 100</param>
/// <“maxOwnerLen">The maximum length that DataShare Owners can be. If 0 or not specified will default to 60</param>
/// <returns>If successful the new ShareStore instance, if not either null or the old ShareStore instance</returns>
public static bool CreateOrOpen(string name = null, int maxNameLen = 0, int maxOwnerLen = 0)

/// Open an existing Share Store optionally using the specified name
/// <“name">The (optional name for the ShareStore (the default name will be used if not specified)</param>
/// <returns>If successful the new ShareStore instance, if not either null or the old ShareStore instance</returns>
public static bool Open(string name = null)

/// Close the currently open Share Store
public static void Close()

/// Does a ShareStore with this name already exist?
/// <“name">The name of the ShareStore to check</param>
/// <returns>True if this ShareStore already exists in memory otherwise false if it does not</returns>
public static bool ShareStoreExists(string name)

Properties

public static readonly string DEFAULT_NAME

/// The Name of this Share Store / Data Share Global area (defaults to MVShareStore Global Area)
public static string Name

/// The current number of users who have access to this Global Area open
public static int Users

/// The number of Headers / Data Shares currently available in the ShareStore

public static int Count

/// Is the DataShare open and ready to receive DataShares?
public static bool isOpen

/// Get the current Date Format value
public static string DateFormat

/// A description of the last error that occurred
public static string LastError

MVConfigEntry Class

The MVConfigEntry class defines an MVConfig Ini / Configuration Entry that can be stored by the MVConfigData class and used by the MVConfigFile class to add and maintain these entries in the Ini / Configuration file you are using. Entries can be either global in scope or they could be in a Section of the file.

Note that Attributes are not currently implemented or supported.

```
///     The type of data held inside an Entry
public enum EntryType {
        /// This type of entry holds a String alphanumeric value.
        STRING,
        /// This type of entry holds an integer numeric value.
        INT,
        /// This type of entry holds a boolean true / false value.
        BOOL,
```

Constructors

```
/// A constructor that creates a default global Entry
/// <”name">The name of the entry</param>
/// <”value">The string (alphanumeric) value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public MVConfigEntry(string name, string value, ArrayList attribs = null)

/// A constructor that creates an Entry in the specified section
/// <”section">The section that is to contain the entry</param>
/// <”name">The name of the entry</param>
/// <”value">The string (alphanumeric) value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public MVConfigEntry(string section, string name, string value, ArrayList attribs = null)

/// A constructor that creates a boolean global Entry
/// <”name">The name of the entry</param>
/// <”value">The boolean value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public MVConfigEntry(string name, bool value, ArrayList attribs = null)

/// A constructor that creates a boolean Entry in a Section
/// <”section">The section that is to contain the entry</param>
/// <”name">The name of the entry</param>
/// <”value">The boolean value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public MVConfigEntry(string section, string name, bool value, ArrayList attribs = null)
```

/// A constructor that creates an Integer global Entry
/// <”name">The name of the entry</param>
/// <”value">The integer (numeric) value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public MVConfigEntry(string name, int value, ArrayList attribs = null)

/// A constructor that creates an Integer Entry in a Section
/// <”section">The section that is to contain the entry</param>
/// <”name">The name of the entry</param>
/// <”value">The integer (numeric) value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public MVConfigEntry(string section, string name, int value, ArrayList attribs = null)

Properties and Functions

/// The name of the section that contains this entry or null if it is global.
public string Section

/// The name of the Section (XML Safe but not required any more).
public string SafeSection

/// A static method that can be used to get a safe section name (not really required any more)
/// <”value">The section name to be converted to an XML Safe name / value</param>
/// <returns>The XML safe value that can be used as a section name</returns>
public static string getSafeSection(string value)

/// The name of the Entry.
public string Name

/// The name of the Entry (XML Safe not really required any more).
public string SafeName

/// The default (string) value of the entry.
public string Value

/// The Boolean value of the entry (defaults to false if it does not exist)
public bool BoolValue

/// The Integer Value of the entry (defaults to 0 if it does not exist)
public int IntValue

/// The Entry Attributes in an ArrayList (ALWAYS RETURNS NULL AT THE MOMENT)
public ArrayList Attribs

/// What type of data does this entry contain?
public EntryType Type

/// Does this Entry contain a String value or not?
/// <returns>True if this entry contains a string (alphanumeric) value, otherwise false</returns>
public bool isString()

/// Does this Entry contain a Boolean value or not?
/// <returns>True if this entry contains a boolean value, otherwise false</returns>
public bool isBoolean()

/// Does this Entry contain an Integer value or not?
/// <returns>True if this entry contains an integer (numeric) value, otherwise false</returns>
public bool isInteger()

/// is this an empty Entry or not? (Boolean types cannot be empty - only false. Integer types might be 0).
/// <returns>True if this entry is empty, otherwise false</returns>
public bool isEmpty()

/// Does this entry have any Attributes defined (WILL ALWAYS RETURN FALSE CURRENTLY)
/// <returns>True if Attributes have been defined for this entry otherwise false (WILL ALWAYS RETURN FALSE CURRENTLY)</returns>
public bool hasAttributes()

/// Static function to convert a Data Item into a Config File Entry instance
/// <”item">The DataItem to convert into a ConfigEntry</param>
/// <returns>The MVConfigEntry instance that is equivalent to the DataItem passed or null if it was not supplied</returns>
public static MVConfigEntry toConfigEntry(DataItem item)

/// Static function to convert a Config Entry into a Data Item instance
/// <”entry">The ConfigEntry to convert into a DataItem</param>
/// <returns>The DataItem instance that is equivalent to the MVConfigEntry passed or null if it was not supplied</returns>
public static DataItem toDataItem(MVConfigEntry entry)

MVConfigData Class

The MVConfigData class is directly derived from the DataStore class so implements all of its capabilities (as well as the DataPack class that it is itself derived from).

It is used to store all the Config File entries in what is effectively a DataPack / DataStore but is accessed using the same API as the MVConfig library class of the same name.

Each Section of the Ini / Configuration file is implemented as a nested DataPack.

Constructors

/// Default ConfigData Constructor that does nothing
public MVConfigData()

/// Create a new ConfigData instance and populate it using the DataItems in the DataPack passed.
/// <”pack">The DataPack to use to populate this ConfigData instance</param>
public MVConfigData(DataPack pack)

/// Constructor to open an existing Config file if it exists or create a new one if it doesn't.
/// Takes the name (and optionally path) of the Config File on disk to open and load or save to.
/// Additionally it can take the application name to use when saving or to check when opening.
/// Note: If the Config file found is in the old MVConfig XML format it will be automatically converted to the new DataStore format.
/// <”filename">The name of the MV Configuration file to open and load or use when saving this configuration data to disk</param>
/// <”appname">The (optional) application name to be used internally when opening this config file to check it is genuine</param>
public MVConfigData(string filename, string appname = null)

File Handling Functions

/// Create a new MV Configuration file in memory. Return the new ConfigData if successful.
/// <returns>True if the file was created successfully otherwise false (The problem will be recorded in getLastError())</returns>
public static MVConfigData NewFile()

/// Open and load the previously specified MV Configuration file on disk. Return true if successful
/// <”filename">The name of the MV configuration file to open and load</param>
/// <”appname">The (optional) Application name to be used internally when opening this config file to check it is genuine</param>
/// <returns>True if the file was opened successfully otherwise false (The problem will be recorded in getLastError())</returns>
public static MVConfigData OpenFile(string filename, string appname = null)

/// Save the current MV Configuration data to the originally specified file on disk.

/// <returns>True if the file was successfully saved. False if it failed (Last error in getLastError())</returns>
public bool saveFile()

Add Functions

/// Add a new string entry into the Global Config Data.
/// <”name">The name used to identify the entry</param>
/// <”value">The string (alphanumeric) value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
/// <returns>True if successful or false if it already exists</returns>
public bool addEntry(string name, string value, ArrayList attribs = null)

/// Add a new string entry into the Section Config Data.
/// <”section">The section that this entry is to go into</param>
/// <”name">The name used to identify the entry</param>
/// <”value">The string (alphanumeric) value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
/// <returns>True if successful or false if it already exists</returns>
public bool addEntry(string section, string name, string value, ArrayList attribs = null)

/// Add a new Boolean entry into the Global Config Data.
/// <”name">The name used to identify the entry</param>
/// <”value">The boolean value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
/// <returns>True if successful or false if it already exists</returns>
public bool addEntry(string name, bool value, ArrayList attribs = null)

/// Add a new Boolean entry into the Section Config Data.
/// <”section">The section that this entry is to go into</param>
/// <”name">The name used to identify the entry</param>
/// <”value">The boolean value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
/// <returns>True if successful or false if it already exists</returns>
public bool addEntry(string section, string name, bool value, ArrayList attribs = null)

/// Add a new Integer entry into the Global Config Data.
/// <”name">The name used to identify the entry</param>
/// <”value">The Integer value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
/// <returns>True if successful or false if it already exists</returns>
public bool addEntry(string name, int value, ArrayList attribs = null)

/// Add a new Integer entry into the Section Config Data.
/// <”section">The section that this entry is to go into</param>

/// <”name">The name used to identify the entry</param>
/// <”value">The integer value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
/// <returns>True if successful or false if it already exists</returns>
public bool addEntry(string section, string name, int value, ArrayList attribs = null)

/// Add a new Section to this Config Data.
/// <”section">The name of the section to be added (required and duplicates not allowed)</param>
/// <returns>The DataPack representing the Section or null if it was not found</returns>
public DataPack addSection(string section)

Clear and Delete Functions

/// Clear all the global Entries in the Config data
/// <returns>The number of global Entries cleared from tne Config data</returns>
public int clearEntries()

/// Clear all the Section Entries in the Config data
/// <returns>The number of Sections cleared from tne Config data</returns>
public int clearSections()

/// Clear all the Entries in the Config data
/// <returns>The number of global Entries and Sections cleared from the Config data</returns>
public int clearAll()

/// Delete the specified Global configuration entry
/// <”name">The name of the global entry to be deleted</param>
/// <returns>True of it was deleted or false if it does not exist</returns>
public bool delEntry(string name)

/// Delete the specified Section configuration entry by name
/// <”section">The Section that owns the entry to be deleted</param>
/// <”name">The name of the entry to be deleted</param>
/// <returns>True of it was deleted or false if it does not exist</returns>
public bool delEntry(string section, string name)

/// Delete the specified Global configuration entry
/// <”index">The (zero based) index of the entry to be deleted and removed</param>
/// <returns>True if the entry was found and deleted otherwise false</returns>
public bool delEntry(int index)

/// Delete the specified Section from the configuration data
/// <”section">the name of the section to be deleted and removed</param>
/// <returns>True if the section was found and deleted otherwise false</returns>
public bool delSection(string section)

Find Functions

// Find the first Entry in the List and make it the current entry
// <returns>True if found otherwise false</returns>
public bool findFirst()

/// Find and return the first Entry in the List and make it the current entry
/// <”entry">Contains the first entry found in the global list</param>
/// <returns>True if found otherwise false</returns>
public bool findFirst(out MVConfigEntry entry)

/// Find and return the first Entry in the specified Section of the List and make it the current entry
/// <”section">The section to find the first record in</param>
/// <returns>True if found otherwise false</returns>
public bool findFirst(string section)

/// Find and return the first Entry in the specified Section
/// <”section">The name of the section to look for entries in</param>
/// <”entry">Contains the first entry in the section</param>
/// <returns>True if found otherwise false</returns>
public bool findFirst(string section, out MVConfigEntry entry)

/// Find the next Entry in the List and make it the current entry.
/// <returns>True if found otherwise false</returns>
public bool findNext()

/// Find and return the next Entry in the List and make it the current entry
/// <”entry">Contains the next entry in the list</param>
/// <returns>True if found otherwise false</returns>
public bool findNext(out MVConfigEntry entry)

/// Find the next Entry in the Section and make it the current entry.
/// <”section">The name of the section to look for entries in</param>
/// <returns>True if found otherwise false</returns>
public bool findNext(string section)

/// Find and return the next Entry in the Section
/// <”section">The name of the section to look for entries in</param>
/// <”entry">Contains the next entry in the section</param>
/// <returns>True if found otherwise false</returns>
public bool findNext(string section, out MVConfigEntry entry)

/// Find and return the name of the first Section found
/// <returns>The first section in the configuration data or null if it does not exist</returns>
public string findFirstSection()

/// Find and return the name of the next Section found
/// <returns>The next section in the configuration data or null if there are no more</returns>
public string findNextSection()

/// Find and return the Section in this ConfigData instance

/// <”section">The name of the Section to find</param>
/// <returns>The Section found in a DataPack or null if it does not exist</returns>
public DataPack findSection(string section)

Get Functions

/// Find the first Entry in the List and make it the current entry.
/// <returns>The first Config Entry if it exists otherwise null</returns>
public new MVConfigEntry getFirst()

/// Find the next Entry in the List and make it the current entry.
/// <returns>The next Config Entry if it exists otherwise null</returns>
public new MVConfigEntry getNext()

/// Get all the Global entries in the configuration data
/// <returns>All the global entries in the configuration data as an array of MVConfigEntry elements</returns>
public MVConfigEntry[] getEntries()

/// Get the value of the current record as an MVConfigEntry element
/// <returns>The value of the current record as an MVConfigEntry element or null if it does not exist or is not defined</returns>
public MVConfigEntry getEntry()

/// Get the value of an existing global entry as an MVConfigEntry element
/// <”name">The name of the entry to find and return</param>
/// <returns>The entry if it exists or null if it was not found</returns>
public MVConfigEntry getEntry(string name)

/// Get and return a Section in this Config Data. Create a new one if it does not exist
/// </summary>
/// <”section">The name of the section to be returned (required and duplicates not allowed)</param>
/// <returns>The DataPack representing the Section. A new DataPack DataItem will be created and added if it does not exist</returns>
public DataPack getSection(string section)

/// Get the specified entry in the specified section in the Config Data
/// <”section">The name of the Section to find the Entry in</param>
/// <”name">The name of the Entry to find</param>
/// <returns>The DataItem found in that Section or null if it does not exist</returns>
public DataItem getSectionEntry(string section, string name)

/// Get all the entries for a particular section as an array of MVConfigEntry elements
/// <”section">The name of the section to get the entries from</param>
/// <returns>All the section entries found as an array of MVConfigEntry elements</returns>
public MVConfigEntry[] getSectionEntries(string section)

/// Get the section of the current entry
/// <returns>The section the current entry is in or null it is a global entry or undefined</returns>

public string getSection()

/// Get the name of the current entry
/// <returns>The name of the current entry or null if it undefined</returns>
public string getName()

/// Get the value of the current entry as a string
/// <returns>A string containing the current value of the entry</returns>
public string getValue()

/// Get the value of the current entry as a boolean
/// <returns>A boolean containing the current value of the entry</returns>
public bool getBoolValue()

/// Get the value of the current entry as an integer
/// <returns>An integer containing the current value of the entry</returns>
public int getIntValue()

/// Get the value of the current entry as a string
/// <returns>The value of the current entry as a string or null if it is not found or undefined</returns>
public string getStringValue()

/// Get any Attributes associated with the current entry
/// <returns>NULL as this is not supported here</returns>
public ArrayList getAttributes()

/// Get the type of the current entry
/// <returns>The EntryType defining the type of value this entry contains</returns>
public MVConfigEntry.EntryType getType()

/// Get the value of any existing global string entry using its name
/// <”name">The name of the global entry to find and return</param>
/// <returns>The string value of the entry if found. Null if it does not exist</returns>
public string getValue(string name)

/// Get the value of any existing string entry in a section using its name
/// <”section">The name of the section to look for the entry</param>
/// <”name">The name of the entry to find</param>
/// <returns>The value of the entry as a string if it is found. Null if it does not exist</returns>
public string getValue(string section, string name)

/// Get the value of any existing global boolean entry using its name
/// <”name">The name of the global entry to find and return</param>
/// <returns>The boolean value of the entry if found. False if it does not exist</returns>
public bool getBoolValue(string name)

/// Get the value of any existing boolean entry in a section using its name
/// <”section">The name of the section to look for the entry</param>
/// <”name">The name of the entry to find</param>
/// <returns>The value of the entry as a boolean if it is found. False if it does not exist</returns>

public bool getBoolValue(string section, string name)

/// Get the value of any existing global integer entry using its name
/// <”name">The name of the global entry to find and return</param>
/// <returns>The numeric value of the entry if found. Zero if it does not exist</returns>
public int getIntValue(string name)

/// Get the value of any existing global integer entry using its name
/// <”section">The name of the section to look for the entry</param>
/// <”name">The name of the global entry to find and return</param>
/// <returns>The numeric value of the entry if found. Zero if it does not exist</returns>
public int getIntValue(string section, string name)

/// Get the value of any existing global string entry using its name
/// <”name">The name of the global entry to find and return</param>
/// <returns>The string value of the entry if found. Null if it does not exist</returns>
public string getStringValue(string name)

/// Get the value of any existing string entry in a section using its name
/// <”section">The name of the section to look for the entry</param>
/// <”name">The name of the entry to find</param>
/// <returns>The value of the entry as a string if it is found. Null if it does not exist</returns>
public new string getStringValue(string section, string name)

/// Get the attributes of any existing global string entry using its name
/// <”name">The name of the global entry to find and return</param>
/// <returns>Any Attributes the specified entry might have or null if it is not found or there are none (THIS WILL ALWAYS RETURN NULL)</returns>
public ArrayList getAttributes(string name)

/// Get the attributes of any existing string entry in a section using its name
/// <”section">The name of the section to look for the entry</param>
/// <”name">The name of the entry to find</param>
/// <returns>Any Attributes the specified entry might have or null if it is not found or there are none (THIS WILL ALWAYS RETURN NULL)</returns>
public ArrayList getAttributes(string section, string name)

Set Functions

/// Add a new Global string entry or update the existing one if found.
/// <”name">The name of the new or existing entry</param>
/// <”value">The string value to give the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public void setEntry(string name, string value, ArrayList attribs = null)

/// Add a new Section string entry or update the existing one if found.
/// <”section">The name of the section that contains the entry</param>
/// <”name">The name of the new or existing entry</param>
/// <”value">The string value to give the entry</param>

/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public void setEntry(string section, string name, string value, ArrayList attribs = null)

/// Add a new Global boolean entry or update the existing one if found.
/// <”name">The name of the new or existing entry</param>
/// <”value">The boolean value to give the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public void setEntry(string name, bool value, ArrayList attribs = null)

/// Add a new Section boolean entry or update the existing one if found.
/// <”section">The name of the section that contains the entry</param>
/// <”name">The name of the new or existing entry</param>
/// <”value">The boolean value to give the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public void setEntry(string section, string name, bool value, ArrayList attribs = null)

/// Add a new Global integer entry or update the existing one if found.
/// <”name">The name of the new or existing entry</param>
/// <”value">The numeric value to give the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public void setEntry(string name, int value, ArrayList attribs = null)

/// Add a new Section integer entry or update the existing one if found.
/// <”section">The name of the section that contains the entry</param>
/// <”name">The name of the new or existing entry</param>
/// <”value">The numeric value to give the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public void setEntry(string section, string name, int value, ArrayList attribs = null)

Exists Functions

/// Does an entry containing the supplied name exist in the Global List:
/// <”name">The name of the entry to search for</param>
/// <returns>True if this entry is in the Global list otherwise false</returns>
public bool entryExists(string name)

/// Does an entry containing the supplied name exist in the specified Section
/// <”section">The Section to search for this entry in</param>
/// <”name">The name of the entry to search for</param>
/// <returns>True if this entry is in the specified Section otherwise false</returns>
public bool entryExists(string section, string name)

/// Does the specified Section exist or not?
/// <”section">The name of the Section to look for</param>
/// <returns>True if that section ecxists, false if it does not</returns>

public bool sectionExists(string section)

Properties

/// The current (last added, found or set) Section.
public string CurrentSection

/// Get the original MVConfig XML Config file. This will be null if no MVConfig file conversion has taken place.
public MVConfig.MVConfigFile XMLConfig

/// Has a Config file been found and loaded?
internal bool Loaded

MVConfigFile Class

The MVConfigFile class is the main class in the MVConfig Library and is used to create, open and save MVConfig Ini / Configuration files.

Additionally it also exposed most of the MVConfigData API meaning that it is normally sufficient to instantiate just an instance of this class in order to access most of the functionality required to maintain MVConfig Ini / Configuration files.

Note that the default file extension for MVConfig files is .mvc but when they are converted to the new DataStore format it will become .mdc.

Constructors

/// Default Constructor that does nothing.
public MVConfigFile()

/// Create a new ConfigFile instance and populate it using the DataItems in the DataPack passed.
/// <”pack">The DataPack to use to populate this ConfigFile instance</param>
public MVConfigFile(DataPack pack)

/// Constructor to open an existing MV Config file if it exists or create a new one if it doesn't.
/// Takes the name (and optionally path) of the MV Config File on disk to open and load or save to.
/// Additionally it can take the application name to use when saving or to check when opening.
/// <”filename">The name of the MV Configuration file to open and load or use when saving this configuration data to disk</param>
/// <”appname">The (optional) application name to be used internally when opening this MV Config file to check it is genuine (NOT CURRENTLY IMPLEMENTED)</param>
public MVConfigFile(string filename, string appname = null)

File Handling Functions

/// Create a new MV Configuration file in memory. Any existing file will be saved first.
/// <”filename">The name of the MV configuration file to use when saving this configuration data to disk</param>
/// <”appname">The (optional) Application name to be used internally when saving or to check when opening (NOT CURRENTLY IMPLEMENTED)</param>
/// <returns>True if the file was created successfully otherwise false (The problem will be recorded in getLastError())</returns>
public bool newFile(string filename, string appname = null)

/// Open and load the previously specified MV Configuration file on disk. Return true if successful. Any existing file will be saved first.
/// <”filename">The name of the MV configuration file to open and load</param>
/// <”appname">The (optional) Application name to be used internally when opening this config file to check it is genuine (NOT CURRENTLY IMPLEMENTED)</param>

/// <returns>True if the file was opened successfully otherwise false (The problem will be recorded in getLastError())</returns>
public bool openFile(string filename, string appname = null)

/// Save the current MV Configuration data to the originally specified file on disk.
/// <returns>True if the file was successfully saved. False if it failed (Last error in getLastError())</returns>
public bool saveFile()

/// Save the current MV Config data to a file disk using the name supplied
/// <”filename">The name of the file on disk to save this configuration data too. If it is null then the Windows Save As dialog will be displayed</param>
/// <”appname">The (optional) Application name to be used internally when opening this config file to check it is genuine (NOT CURRENTLY IMPLEMENTED)</param>
/// <”overwrite">Optional and set by default to false. Set to true to overwrite any existing file. If false then this will fail if the file already exists</param>
/// <returns>True if the file was successfully saved. False if it failed (Last error in getLastError())</returns>
public bool saveFileAs(string filename, string appname = null, bool overwrite = false)

/// Close the current MV Config file, saving it to disk if needed.
public void close()

/// Set and return the name of this Config file
public static string setFilename(string filename)

Add Functions

/// Add a new string entry into the Global Config Data.
/// <”name">The name used to identify the entry</param>
/// <”value">The string (alphanumeric) value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
/// <returns>True if successful or false if it already exists</returns>
public bool addEntry(string name, string value, ArrayList attribs = null)

/// Add a new string entry into the Section Config Data.
/// <”section">The section that this entry is to go into</param>
/// <”name">The name used to identify the entry</param>
/// <”value">The string (alphanumeric) value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
/// <returns>True if successful or false if it already exists</returns>
public bool addEntry(string section, string name, string value, ArrayList attribs = null)

/// Add a new Boolean entry into the Global Config Data.
/// <”name">The name used to identify the entry</param>
/// <”value">The boolean value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>

/// <returns>True if successful or false if it already exists</returns>
public bool addEntry(string name, bool value, ArrayList attribs = null)

/// Add a new Boolean entry into the Section Config Data.
/// <”section">The section that this entry is to go into</param>
/// <”name">The name used to identify the entry</param>
/// <”value">The boolean value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
/// <returns>True if successful or false if it already exists</returns>
public bool addEntry(string section, string name, bool value, ArrayList attribs = null)

/// Add a new Integer entry into the Global Config Data.
/// <”name">The name used to identify the entry</param>
/// <”value">The Integer value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
/// <returns>True if successful or false if it already exists</returns>
public bool addEntry(string name, int value, ArrayList attribs = null)

/// Add a new Integer entry into the Section Config Data.
/// <”section">The section that this entry is to go into</param>
/// <”name">The name used to identify the entry</param>
/// <”value">The integer value of the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
/// <returns>True if successful or false if it already exists</returns>
public bool addEntry(string section, string name, int value, ArrayList attribs = null)

/// Add a new Section to this Config Data.
/// <”section">The name of the section to be added (required and duplicates not allowed)</param>
/// <returns>The DataPack representing the Section or null if it was not found</returns>
public DataPack addSection(string section)

Clear and Delete Functions

/// Clear all the global Entries in the Config data
/// <returns>The number of global Entries cleared from tne Config data</returns>
public int clearEntries()

/// Clear all the Section Entries in the Config data
/// <returns>The number of Sections cleared from tne Config data</returns>
public int clearSections()

/// Clear all the Entries in the Config data
/// <returns>The number of global Entries and Sections cleared from the Config data</returns>
public int clearAll()

/// Delete the specified Global configuration entry

/// <”name">The name of the global entry to be deleted</param>
/// <returns>True of it was deleted or false if it does not exist</returns>
public bool delEntry(string name)

/// Delete the specified Section configuration entry by name
/// <”section">The Section that owns the entry to be deleted</param>
/// <”name">The name of the entry to be deleted</param>
/// <returns>True of it was deleted or false if it does not exist</returns>
public bool delEntry(string section, string name)

/// Delete the specified Global configuration entry
/// <”index">The (zero based) index of the entry to be deleted and removed</param>
/// <returns>True if the entry was found and deleted otherwise false</returns>
public bool delEntry(int index)

/// Delete the specified Section from the configuration data
/// <”section">the name of the section to be deleted and removed</param>
/// <returns>True if the section was found and deleted otherwise false</returns>
public bool delSection(string section)

Find Functions

// Find the first Entry in the List and make it the current entry
// <returns>True if found otherwise false</returns>
public bool findFirst()

/// Find and return the first Entry in the List and make it the current entry
/// <”entry">Contains the first entry found in the global list</param>
/// <returns>True if found otherwise false</returns>
public bool findFirst(out MVConfigEntry entry)

/// Find and return the first Entry in the specified Section of the List and make it the current entry
/// <”section">The section to find the first record in</param>
/// <returns>True if found otherwise false</returns>
public bool findFirst(string section)

/// Find and return the first Entry in the specified Section
/// <”section">The name of the section to look for entries in</param>
/// <”entry">Contains the first entry in the section</param>
/// <returns>True if found otherwise false</returns>
public bool findFirst(string section, out MVConfigEntry entry)

/// Find the next Entry in the List and make it the current entry.
/// <returns>True if found otherwise false</returns>
public bool findNext()

/// Find and return the next Entry in the List and make it the current entry
/// <”entry">Contains the next entry in the list</param>
/// <returns>True if found otherwise false</returns>
public bool findNext(out MVConfigEntry entry)

/// Find the next Entry in the Section and make it the current entry.
/// <”section">The name of the section to look for entries in</param>
/// <returns>True if found otherwise false</returns>
public bool findNext(string section)

/// Find and return the next Entry in the Section
/// <”section">The name of the section to look for entries in</param>
/// <”entry">Contains the next entry in the section</param>
/// <returns>True if found otherwise false</returns>
public bool findNext(string section, out MVConfigEntry entry)

/// Find and return the name of the first Section found
/// <returns>The first section in the configuration data or null if it does not exist</returns>
public string findFirstSection()

/// Find and return the name of the next Section found
/// <returns>The next section in the configuration data or null if there are no more</returns>
public string findNextSection()

/// Find and return the Section in this ConfigData instance
/// <”section">The name of the Section to find</param>
/// <returns>The Section found in a DataPack or null if it does not exist</returns>
public DataPack findSection(string section)

Get Functions

/// Find the first Entry in the List and make it the current entry.
/// <returns>The first Config Entry if it exists otherwise null</returns>
public new MVConfigEntry getFirst()

/// Find the next Entry in the List and make it the current entry.
/// <returns>The next Config Entry if it exists otherwise null</returns>
public new MVConfigEntry getNext()

/// Get all the Global entries in the configuration data
/// <returns>All the global entries in the configuration data as an array of MVConfigEntry elements</returns>
public MVConfigEntry[] getEntries()

/// Get the value of the current record as an MVConfigEntry element
/// <returns>The value of the current record as an MVConfigEntry element or null if it does not exist or is not defined</returns>
public MVConfigEntry getEntry()

/// Get the value of an existing global entry as an MVConfigEntry element
/// <”name">The name of the entry to find and return</param>
/// <returns>The entry if it exists or null if it was not found</returns>
public MVConfigEntry getEntry(string name)

/// Get the value of an existing entry in a section as an MVConfigEntry element
/// <”section">The name of the section to search for the entry in</param>
/// <”name">The name of the entry to find and return</param>
/// <returns>The entry if it exists or null if it was not found</returns>
public MVConfigEntry getEntry(string section, string name)

/// Get the specified entry in the specified section in the Config Data
/// <”section">The name of the Section to find the Entry in</param>
/// <”name">The name of the Entry to find</param>
/// <returns>The DataItem found in that Section or null if it does not exist</returns>
public DataItem getSectionEntry(string section, string name)

/// Get all the entries for a particular section as an array of MVConfigEntry elements
/// <”section">The name of the section to get the entries from</param>
/// <returns>All the section entries found as an array of MVConfigEntry elements</returns>
public MVConfigEntry[] getSectionEntries(string section)

/// Get the names of all the sections as an array of strings
/// <returns>A string array containing the names of all the sections or null if there are none</returns>
public string[] getSectionNames()

/// Get the section of the current entry
/// <returns>The section the current entry is in or null it is a global entry or undefined</returns>
public string getSection()

/// Get the name of the current entry
/// <returns>The name of the current entry or null if it undefined</returns>
public string getName()

/// Get the value of the current entry as a string
/// <returns>A string containing the current value of the entry</returns>
public string getValue()

/// Get the value of any existing global boolean entry using its name
/// <”name">The name of the global entry to find and return</param>
/// <returns>The boolean value of the entry if found. False if it does not exist</returns>
public bool getBoolValue(string name)

/// Get the value of any existing boolean entry in a section using its name
/// <”section">The name of the section to look for the entry</param>
/// <”name">The name of the entry to find</param>
/// <returns>The value of the entry as a boolean if it is found. False if it does not exist</returns>
public bool getBoolValue(string section, string name)

/// Get the value of any existing global integer entry using its name
/// <”name">The name of the global entry to find and return</param>
/// <returns>The numeric value of the entry if found. Zero if it does not exist</returns>
public int getIntValue(string name)

/// Get the value of any existing global integer entry using its name

/// <”section">The name of the section to look for the entry</param>
/// <”name">The name of the global entry to find and return</param>
/// <returns>The numeric value of the entry if found. Zero if it does not exist</returns>
public int getIntValue(string section, string name)

/// Get the value of any existing global string entry using its name
/// <”name">The name of the global entry to find and return</param>
/// <returns>The string value of the entry if found. Null if it does not exist</returns>
public string getStringValue(string name)

/// Get the value of any existing string entry in a section using its name
/// <”section">The name of the section to look for the entry</param>
/// <”name">The name of the entry to find</param>
/// <returns>The value of the entry as a string if it is found. Null if it does not exist</returns>
public new string getStringValue(string section, string name)

/// Get the attributes of any existing global string entry using its name
/// <”name">The name of the global entry to find and return</param>
/// <returns>Any Attributes the specified entry might have or null if it is not found or there are none (THIS WILL ALWAYS RETURN NULL)</returns>
public ArrayList getAttributes(string name)

/// Get the attributes of any existing string entry in a section using its name
/// <”section">The name of the section to look for the entry</param>
/// <”name">The name of the entry to find</param>
/// <returns>Any Attributes the specified entry might have or null if it is not found or there are none (THIS WILL ALWAYS RETURN NULL)</returns>
public ArrayList getAttributes(string section, string name)

Set Functions

/// Add a new Global string entry or update the existing one if found.
/// <”name">The name of the new or existing entry</param>
/// <”value">The string value to give the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public void setEntry(string name, string value, ArrayList attribs = null)

/// Add a new Section string entry or update the existing one if found.
/// <”section">The name of the section that contains the entry</param>
/// <”name">The name of the new or existing entry</param>
/// <”value">The string value to give the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public void setEntry(string section, string name, string value, ArrayList attribs = null)

/// Add a new Global boolean entry or update the existing one if found.
/// <”name">The name of the new or existing entry</param>
/// <”value">The boolean value to give the entry</param>

/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public void setEntry(string name, bool value, ArrayList attribs = null)

/// Add a new Section boolean entry or update the existing one if found.
/// <”section">The name of the section that contains the entry</param>
/// <”name">The name of the new or existing entry</param>
/// <”value">The boolean value to give the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public void setEntry(string section, string name, bool value, ArrayList attribs = null)

/// Add a new Global integer entry or update the existing one if found.
/// <”name">The name of the new or existing entry</param>
/// <”value">The numeric value to give the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public void setEntry(string name, int value, ArrayList attribs = null)

/// Add a new Section integer entry or update the existing one if found.
/// <”section">The name of the section that contains the entry</param>
/// <”name">The name of the new or existing entry</param>
/// <”value">The numeric value to give the entry</param>
/// <”attribs">Optional Attributes in an ArrayList to be added to the entry (NOT CURRENTLY IMPLEMENTED)</param>
public void setEntry(string section, string name, int value, ArrayList attribs = null)

Exists Functions

/// Does an entry containing the supplied name exist in the Global List:
/// <”name">The name of the entry to search for</param>
/// <returns>True if this entry is in the Global list otherwise false</returns>
public bool entryExists(string name)

/// Does an entry containing the supplied name exist in the Global List or in a Section
/// <param name="entry">The entry to search for</param>
/// <returns>True if this entry is in the Global list or Section otherwise false</returns>
public bool entryExists(MVConfigEntry entry)

/// Does an entry containing the supplied name exist in the specified Section
/// <”section">The Section to search for this entry in</param>
/// <”name">The name of the entry to search for</param>
/// <returns>True if this entry is in the specified Section otherwise false</returns>
public bool entryExists(string section, string name)

/// Does the specified Section exist or not?
/// <”section">The name of the Section to look for</param>
/// <returns>True if that section ecxists, false if it does not</returns>
public bool sectionExists(string section)

Properties

/// Get all the Configuration data as a ConfigData / DataStore instance
/// Note: Return type is now MVShareStore.ConfigData not MVConfig.MVConfigData
/// This can be used in a foreach statement if desired.
/// <returns>All the Configuration data as a ConfigData / DataStore instance</returns>
public MVConfigData getData()

/// Set all the Configuration data using a ConfigData instance
/// Note: Return and parameter type is now MVShareStore.ConfigData not MVConfig.MVConfigData
/// <returns>All the Configuration data just set as a ConfigData instance</returns>
public MVConfigData setData(MVConfigData value)

/// Has an MVConfig / DataStore file been found and loaded?
/// <returns>True if a MVConfig / DataStore file has been succcessfully loaded</returns>
public bool getLoaded()

/// Get the number of entries in the Config data file
/// <returns>The number of entries currently in the config data</returns>
public int getCount()

/// Has the data in this configuration file been changed?
/// <returns>True if the configuration needs has changed and needs saving otherwise false</returns>
public bool getDirty()

/// Return the version number of this Library
/// <returns>The version of this Library as a string (unlike the original API which had it as a double)</returns>
public static string getVersion()

/// Get the name of this Config file
public string getFilename()

Settings Class

There are a few settings that can be accessed in the Settings class. These settings are automatically saved and then read by the Library.

/// The Version of the Library as a string
public static string Version

/// The maximum length that DataStore Names can be in this ShareStore (defaults to 90)
public static int MaxNameLength

/// The maximum length that DataStore Owners can be in this ShareStore (defaults to 60)
public static int MaxOwnerLength

/// How many MB of memory should be reserved for DataShare Headers (defaults to enough for over 5,000 DataShare Headers)
public static int MBHeaderMemory

/// Hide (or show) Client & Server Message Notifications to the users?
public static bool HideNotifications

/// Enable everything assigned to Data.LastError to be held in an Error History List
public static bool ErrorHistoryEnabled

/// "The number of entries allowed in the Error History List before the oldest start to be retired
public static int ErrorHistoryDepth

Forms

The library also provides a small set of simple Windows dialogs / forms that can be used to assist in developing applications if desired. These are available via the Data Class Static API.

Each form's function, it's constructor, its public properties and Dialog Results are detailed here.

AddHeaderItem

This form can be used to create and add new Header Meta Data Items in much the same way as the AddDataItem form below except that in this case the Data Items created by this form are added to the DataHeader properties in the relevant DataShare or DataStore.

/// Create a new AddHeaderItem Form which can be used, as its name implies, to add new HeaderItems to DataShares.
/// <param name="header">The DataShare Data Header to add the Items to</param>
public AddHeaderItem(DataHeader header = null)

/// The resulting DataHeader instance populated with the new Data Items
Notes:

- If a null DataHeader is passed to this form and the form is cancelled out of then this will be null
- If a null DataHeader is passed to this form and Data Items are added then this will be a new DataHeader instance containing them
- If an existing DataHeader is passed to this form then this will contain that DataHeader even if the form is cancelled out of
- If an existing DataHeader is passed to this form and items added then they will added to any items that it already contains

public DataHeader DataHeader

Returns:
Accept = DialogResult.OK
Cancel = DialogResult.Cancel

AddDataItem

This form can be used to add new DataItems to a DataPack / DataShare / DataStore or to edit an existing Data Item if one is passed as a parameter to the form's constructor.

/// Create a new AddDataItem Form which can be used, as its name implies, to add new DataItems to DataPacks / DataShares / DataStores.
/// It can also be used to Edit / Modify and Insert Data Items.
/// <"dataPack">The DataPack / DataShare / DataStore to add the Items to.</param>
/// <"item">If a Data Item is supplied here then it will be displayed ready for editing / modifying</param>
/// <"index">If an index value of 0 or higher is specified then that is the index used to insert it into the DataPack / DataShare / DataStore</param>
public AddDataItem(DataPack dataPack, DataItem item = null, int index = -1)

/// The DataPack of Data Items that have been added using this form
/// </summary>
public DataPack DataItems

/// The current Data Item (last added)
public DataItem DataItem

Returns:
Accept = DialogResult.OK
Cancel = DialogResult.Cancel

CreateDataShare

This form can be used to either create a new DataShare or convert the supplied DataPack / DataShare or DataStore into a DataShare using the information such as Name, Owner, Private and Readonly entered into the form.

/// Create a DataShare, getting the DataShare specific information required such as Name, Owner, Private, Readonly
/// <“data">The DataPack / DataShare or DataStore to make into a DataShare and add this information to or if null then a new DataShare will be created</param>
public CreateDataShare(DataPack data = null)

/// Either the DataPack / DataShare / DataStore supplied made into a DataShare or a new DataShare
public DataShare DataShare

Returns:
Accept = DialogResult.OK
Cancel = DialogResult.Cancel

CreateDataStore

Create a new DataStore getting its Name and Path information or modify a DataPack or DataShare to make it a DataStore. It can also open and modify an existing DataStore.

/// Create a new DataStore or change a DataPack / DataShare into a DataStore.
/// <”data">The optional DataPack / DataShare to make into a DataStore</param>
/// <”open">Should an existing DataStore be opened and used or not?</param>
public CreateDataStore(DataPack data = null, bool open = false)

/// The resulting new DataStore or modified DataPack or DataShare as a DataStore
public DataStore DataStore

Returns:
Accept = DialogResult.OK
Cancel = DialogResult.Cancel

EditDataItems

Edit all the Data Items in a DataPack / DataShare or DataStore, adding, editing, deleting and inserting them as desired. If it is a DataShare it can also Post / Repost it after it has been edited.

/// Edit the Data Items in the supplied DataPack / DataShare or DataStore. Add, Delete, Insert Data Items. Modify the Data Items already there. Then Save, Post or Repost the DataPack / DataShare or DataStore after it has been modified.
/// <”data">The DataPack / DataShare or DataStore to be edited</param>
public EditDataItems(DataPack data)

/// The Data Pack to list / edit the Data Items in
public DataPack DataPack

/// The Data Share to list / edit the Data Items in
public DataShare DataShare

/// The Data Store to list / edit the Data Items in
public DataStore DataStore

Returns:
Accept = DialogResult.OK
Cancel = DialogResult.Cancel

ListDataItems

List all the Data Items in a DataPack / DataShare or DataStore (basically a read-only version of EditDataItems)

/// List the Data Items in the supplied DataPack / DataShare or DataStore (basically a read only version of the EditDataItems form).
/// <param name="data">The DataPack / DataShare or DataStore to be listed</param>
public ListDataItems(DataPack data)

/// The Data Pack to list the Data Items in
public DataPack DataPack

/// The Data Share to list the Data Items in
public DataShare DataShare

/// The Data Store to list the Data Items in
public DataStore DataStore

Returns:
Accept = DialogResult.OK
Cancel = DialogResult.Cancel

ListDataShares

List all the DataShares that have been Posted / Published to this Share Store and can therefore be accessed by any interested process. The DataShares in the List show their current details and status. The Shares in the List can also optionally be filtered so that only the desired Shares are listed and / or they can be sorted so that they appear in a particular order in the List.

/// List all the (Headers of the) Data Shares that have been posted / published and are available in this Share Store to be opened by other processes. These can be sorted and / or filtered as desired.
/// <”shares">The Data Share Headers to be listed. If null then the current list will be determined and used</param>
/// <”allowEdit">Can DataShares in the List be edited and can the list itself be edited? Note: Only DataShares added by this process are allowed to be deleted from the list</param>
/// <”filters">What filters to apply to the Data Shares listed (using a ShareList.Filters instance) or null for all Data Shares</param>
/// <”sortBy">What field to sort the Data Share list on (using a ShareList.Sort instance) or null for unsorted</param>
public ListDataShares(List<Header> shares = null, bool allowEdit = false, ShareList.Filters filters = null, ShareList.Sort sortBy = null)

/// The Data Share Headers in the current List
public List<Header> Shares

Returns:
Accept = DialogResult.OK

ListErrorMessages

List all the Error Messages in the error message history (if it has been enabled).

/// List all the error messages in the error message history
public ListErrorMessages(List<string>errors)

MergeSelect

Select which DataPack / DataShare or DataStore to merge with the DataPack / DataShare or DataStore passed to the form. New DataPacks can be created and DataStores can be loaded from disk. Merged DataPacks can also be edited and / or saved as DataStores to disk.

/// Select a DataPack / DataShare or DataStore to merge with the supplied DataPack / DataShare or DataStore (which can be opened from a DataStore on disk or a new DataPack created).
/// The merged DataPack / DataShare or DataStore can optionally be viewed, edited or saved to a DataStore on disk.
/// <”data">The DataPack / DataShare or DataStore you want to merge another DataPack / DataShare or DataStore with</param>
public MergeSelect(DataPack data)

/// The original DataPack / DataShare or DataStore supplied and later merged with the selected DataPack / DataShare or DataStore.
public DataPack DataPack

/// The selected DataPack / DataShare or DataStore merged with the original DataPack / DataShare or DataStore.
public DataPack DataPack2

Returns:
Accept = DialogResult.OK
Cancel = DialogResult.Cancel

SetShareFilters

Select which Data Shares to include / exclude and display in the Share List.

/// Select which Data Shares to include / exclude and display in the Share List.
/// <”filters">The optional ShareList.Filters set of DataShare filters to preload the form with</param>
public SetShareFilters(ShareList.Filters filters = null)

/// The current set of ShareList Filters (can be passed to ListDataShares form)
public ShareList.Filters Filters

Returns:
Accept = DialogResult.OK
Cancel = DialogResult.Cancel

SortShareList

Select which field you want to sort the Shares in the Share List and whether to sort in ascending order (the default) or descending order.

/// Select which field you want to sort the Shares in the Share List and whether to sort in ascending order (the default) or descending order
/// <”sort">The optional ShareList.Sort instance containing an existing sort order to prefill the form with</param>
public SortShareList(ShareList.Sort sort = null)

/// The ShareList Sort instance detailing the sort order for the Shares in the List (can be passed to ListDataShares form)
public ShareList.Sort Sort

Returns:
Accept = DialogResult.OK
Cancel = DialogResult.Cancel

Settings

There are a few settings in the MVShareStore Library that might be potentially useful to processes or programs using the Library.

Settings Functions in Data Class

/// Read the Library Settings from the DataStore on disk. If it is not there the default settings will be used
/// <returns>True if successful otherwise false</returns>
public static bool ReadSettings()

/// Save the current Library Settings to a DataStore on disk. Create a new one if it does not already exist
/// <returns>True if successful otherwise false</returns>
public static bool SaveSettings()

Settings Values in Settings Class

/// The Version of the Library as a string
/// </summary>
public static string Version

/// The maximum length that DataStore Names can be in this ShareStore (defaults to 90)
public static int MaxNameLength

/// The maximum length that DataStore Owners can be in this ShareStore (defaults to 60)
public static int MaxOwnerLength

/// How many MB of memory should be reserved for DataShare Headers (defaults to enough for over 5,000 DataShare Headers)
public static int MBHeaderMemory

/// Hide (or show) Client & Server Message Notifications to the users?
public static bool HideNotifications