Scientific DataSet for C++ User's Guide

Getting started

C++ interface to Scientific DataSet is defined in single include file named sds.h. Installer package places it in %ProgramFiles%\ScientificDataSet\Include folder and adds this folder to INCLUDE environment variable.

Steps to create a simple C++ project working with Scientific DataSet are listed below.

1. Create Managed C++ project.

2. You need to set up additional include and reference directories. Open Tool\Options dialog window. Select Projects and Solutions\VC++ Directories in the left panel. Add <ProgramFiles>\ScientificDataSet\Include to Include files, add <ProgramFiles>\Reference Assemblies\Microsoft\Research\ScientificDataSet to Reference files where <ProgramFiles> is path to Program Files on your computer.

3. Add following lines to you C++ file.

   ```
   #include<sds.h>
   #using<Microsoft.Research.Science.Data.dll>
   using namespace CppWrapper;
   ```

Class Reference

All classes are located in CppWrapper namespace.

CppDataSet class methods

Create or open dataset method group.

Opens or creates DataSet with specified URI.

```
static CppDataSet * Open(const char * cstr);
```

For more detailed documentation, look in managed DataSet class Open method.

Parameters

URI – DataSet URI.

Return

CppDataset *

Add variable method group.

Adds variable to specified dataset

```
CppVariable * Add(DataType type, const char * name, const char * dim);
```

```
CppVariable * Add(DataType type, const char * name, char ** dims, int
dimsNumber);
```

Parameters

type – type of the variable to create

name – name of the variable to create

dim(or dims) – variable dimension(s)

dimsNumber – number of variable dimensions(rank of the variable)

Returns

CppVariable *

Value method group.

Returns variable element with specified index

```
VariableElement Value(const char * name, int index);
VariableElement Value(const char * name, int index1, int index2);
VariableElement Value(const char * name, int * indexArr);
```

Parameters

name – name of the variable

index(or index1, index2, or indexArr) – variable element index

VariableElement type then can be assigned to any of DataSet supported types. Also, value of any supported type can be assigned to VariableElement. If variable element type doen't match value type, assigned to it, or if value type doesn't match variable element type, assigned to it, exception would be thown.

Put array method group.

Writes data into array variable.

```
void PutShorts(const char * name, short * values, int * shape);
void PutInts(const char * name, int * values, int * shape);
void PutLongLongs(const char * name,long long * values, int * shape);
void PutUnsignedShorts(const char * name, unsigned short * values, int * shape);
void PutUnsignedInts(const char * name, unsigned int * values, int * shape);
void PutUnsignedLongLongs(const char * name, unsigned long long * values, int * shape);
void PutFloats(const char * name, float * values, int * shape);
void PutDoubles(const char * name, double * values, int * shape);
void PutStrings(const char * name, char ** values, int * shape);
void PutDateTimes(const char * name, time_t * values, int * shape);
void PutBytes(const char * name, byte * values, int * shape);
void PutSignedBytes(const char * name, sbyte * values, int * shape);
void PutBooleans(const char * name, bool * values, int * shape);
```

Parameters

name – variable name

values – data to write into variable. Array "values" is flat (multidimensional array, C-Style stretched to flat array).

shape – shape of data values (shape of multidimensional array, stored in “values” flat array)

PutStrings uses char** instead of array of strings.

PutDateTimes uses time_t * instead of array of DateTime.

If specified name is not found, exception would be thrown.

If specified variable has another type, exception would be thrown.

Get array method group.

Reads all data from array variable.

```
char ** GetStrings(const char * name, int ** shape);
short * GetShorts(const char * name, int ** shape);
int * GetInts(const char * name, int ** shape);
long long * GetLongLongs(const char * name, int ** shape);
unsigned short * GetUnsignedShorts(const char * name, int ** shape);
unsigned int * GetUnsignedInts(const char * name, int ** shape);
unsigned long long * GetUnsignedLongLongs(const char * name, int ** shape);
float * GetFloats(const char * name, int ** shape);
double * GetDoubles(const char * name, int ** shape);
time_t * GetDateTimes(const char * name, int ** shape);
byte * GetBytes(const char * name, int ** shape);
sbyte * GetSignedBytes(const char * name, int ** shape);
bool * GetBooleans(const char * name, int ** shape);
```

Parameters

name – variable name

shape – shape of variable. Should be null to prevent memory leak. It will be filled with variable shape inside method;

Reads specified data from array variable.

```
char ** GetStrings(const char * name, int * shape, int * origin);
short * GetShorts(const char * name, int * shape, int * origin);
unsigned int * GetUnsignedInts(const char * name, int * shape, int * origin);
long long * GetLongLongs(const char * name, int * shape, int * origin);
unsigned short * GetUnsignedShorts(const char * name, int * shape, int * origin);
bool * GetBooleans(const char * name, int * shape, int * origin);
sbyte * GetSignedBytes(const char * name, int * shape, int * origin);
byte * GetBytes(const char * name, int * shape, int * origin);
time_t * GetDateTimes(const char * name, int * shape, int * origin);
double * GetDoubles(const char * name, int * shape, int * origin);
float * GetFloats(const char * name, int * shape, int * origin);
unsigned long long * GetUnsignedLongLongs(const char * name, int * shape, int * origin);
int * GetInts(const char * name, int * shape, int * origin);
```

Parameters

name – variable name

shape – shape of values to get

origin – starting multidimensional index

GetStrings uses char** instead of array of strings.

GetDateTimes uses time_t * instead of array of DateTime.

If specified name is not found, exception would be thrown.

If specified variable has another type, exception would be thrown.

Get scalar method group

Get value of scalar variable.

```
int GetInt(const char * name, int * indexArr);
short GetShort(const char * name, int * indexArr);
long long GetLongLong(const char * name, int * indexArr);
unsigned short GetUnsignedShort(const char * name, int * indexArr);
unsigned int GetUnsignedInt(const char * name, int * indexArr);
unsigned long long GetUnsignedLongLong(const char * name, int * indexArr);
float GetFloat(const char * name, int * indexArr);
double GetDouble(const char * name, int * indexArr);
char * GetString(const char * name, int * indexArr);
time_t GetDateTime(const char * name, int * indexArr);
byte GetByte(const char * name, int * indexArr);
sbyte GetSignedByte(const char * name, int * indexArr);
bool GetBoolean(const char * name, int * indexArr);
```

Parameters

name – variable name

indexArr – multidimensional index of value to get

GetString uses char* instead of string.

GetDateTime uses time_t instead of DateTime.

If specified name is not found, exception would be thrown.

If specified variable has another type, exception would be thrown.

Get value of scalar one dimensional variable.

```
int GetInt(const char * name, int index);
short GetShort(const char * name, int index);
long long GetLongLong(const char * name, int index);
unsigned short GetUnsignedShort(const char * name, int index);
unsigned int GetUnsignedInt(const char * name, int index);
unsigned long long GetUnsignedLongLong(const char * name, int index);
float GetFloat(const char * name, int index);
double GetDouble(const char * name, int index);
char * GetString(const char * name, int index);
```

```
time_t GetDateTime(const char * name, int index);
byte GetByte(const char * name, int index);
sbyte GetSignedByte(const char * name, int index);
bool GetBoolean(const char * name, int index);
```

name – variable name

index – index of value to get

GetString uses char* instead of string.

GetDateTime uses time_t instead of DateTime.

If specified name is not found, exception would be thrown.

If specified variable has another type, exception would be thrown.

Put scalar method group

Put value to scalar variable.

```
void PutShort(const char * name, short value, int * indexArr);
void PutInt(const char * name, int value, int * indexArr);
void PutLongLong(const char * name, long long value, int * indexArr);
void PutUnsignedShort(const char * name, unsigned short value, int * indexArr);
void PutUnsignedInt(const char * name, unsigned int value, int * indexArr);
void PutUnsignedLongLong(const char * name, unsigned long long value, int * indexArr);
void PutFloat(const char * name, float value, int * indexArr);
void PutDouble(const char * name, double value, int * indexArr);
void PutString(const char * name, char * value, int * indexArr);
void PutDateTime(const char * name, time_t value, int * indexArr);
void PutByte(const char * name, byte value, int * indexArr);
void PutSignedByte(const char * name, sbyte value, int * indexArr);
void PutBoolean(const char * name, bool value, int * indexArr);
```

Parameters

name – variable name

value – value to put

indexArr – multidimensional index of value to put

PutString uses char* instead of string.

PutDateTime uses time_t instead of DateTime.

If specified name is not found, exception would be thrown.

If specified variable has another type, exception would be thrown.

Put value to scalar one dimensional variable.

```
void PutShort(const char * name, short value, int index);
void PutInt(const char * name, int value, int index);
void PutLongLong(const char * name, long long value, int index);
void PutUnsignedShort(const char * name, unsigned short value, int index);
void PutUnsignedInt(const char * name, unsigned int value, int index);
void PutUnsignedLongLong(const char * name, unsigned long long value, int index);
void PutFloat(const char * name, float value, int index);
void PutDouble(const char * name, double value, int index);
void PutString(const char * name, char * value, int index);
void PutDateTime(const char * name, time_t value, int index);
void PutByte(const char * name, byte value, int index);
void PutSignedByte(const char * name, sbyte value, int index);
void PutBoolean(const char * name, bool value, int index);
```

Parameters

name – variable name

value – value to put

index – index of value to put

PutString uses char* instead of string.

PutDateTime uses time_t instead of DateTime.

If specified name is not found, exception would be thrown.

If specified variable has another type, exception would be thrown.

Append method group

Append value to variable default dimension.

```
void AppendShort(const char * name, short value);
void AppendInt(const char * name, int value);
void AppendLongLong(const char * name, long long value);
void AppendUnsignedShort(const char * name, unsigned short value);
void AppendUnsignedInt(const char * name, unsigned int value);
void AppendUnsignedLongLong(const char * name, unsigned long long value);
void AppendFloat(const char * name, float value);
void AppendDouble(const char * name, double value);
void AppendString(const char * name, char * value);
void AppendDateTime(const char * name, time_t value);
void AppendByte(const char * name, byte value);
void AppendSignedByte(const char * name, sbyte value);
void AppendBoolean(const char * name, bool value);
```

Parameters

name – variable name

value – value to append

If specified name is not found, exception would be thrown.

If specified variable has another type, exception would be thrown.

Append value to variable specified dimension.

```
void AppendShort(const char * name, short value, const char * dim);
void AppendInt(const char * name, int value, const char * dim);
void AppendLongLong(const char * name, long long value, const char * dim);
void AppendUnsignedShort(const char * name, unsigned short value, const char * dim);
void AppendUnsignedInt(const char * name, unsigned int value, const char * dim);
void AppendUnsignedLongLong(const char * name, unsigned long long value, const char * dim);
void AppendFloat(const char * name, float value, const char * dim);
void AppendDouble(const char * name, double value, const char * dim);
void AppendString(const char * name, char * value, const char * dim);
void AppendDateTime(const char * name, time_t value, const char * dim);
void AppendByte(const char * name, byte value, const char * dim);
void AppendSignedByte(const char * name, sbyte value, const char * dim);
void AppendBoolean(const char * name, bool value, const char * dim);
```

Parameters

name – variable name

dim – dimension to append to

value – value to append

If specified name is not found, exception would be thrown.

If specified variable has another type, exception would be thrown.

Append values to variable default dimension.

```
void AppendShorts(const char * name, short * values, int valuesRank , int * shape);
void AppendInts (const char * name, int * values, int valuesRank , int * shape);
void AppendLongLongs(const char * name, long long * values, int valuesRank , int * shape);
void AppendUnsignedShorts(const char * name, unsigned short * values, int valuesRank , int *
shape);
void AppendUnsignedInts(const char * name, unsigned int * values, int valuesRank , int * shape);
void AppendUnsignedLongLongs(const char * name, unsigned long long * values, int valuesRank , int
* shape);
void AppendFloats(const char * name, float * values, int valuesRank , int * shape);
void AppendDoubles(const char * name, double * values, int valuesRank , int * shape);
void AppendStrings(const char * name, char ** values, int valuesRank , int * shape);
void AppendDateTimes(const char * name, time_t * values, int valuesRank , int * shape);
void AppendBytes(const char * name, byte * values, int valuesRank , int * shape);
void AppendSignedBytes(const char * name, sbyte * values, int valuesRank , int * shape);
void AppendBooleans(const char * name, bool * values, int valuesRank , int * shape);
```

Parameters

name – variable name

values – values to append. Array “values” is flat (multidimensional array, C-Style stretched to flat array).

valuesRank – rank of values. Values must have rank equal or one less than variable rank

shape – shape of the multidimensional value to append

If specified name is not found, exception would be thrown.

If specified variable has another type, exception would be thrown.

Append values to variable dimension.

```
void AppendShorts(const char * name, short * values, int valuesRank , int * shape, const char *
dim);
void AppendInts (const char * name, int * values, int valuesRank , int * shape, const char *
dim);
void AppendLongLongs(const char * name, long long * values, int valuesRank , int * shape, const
char * dim);
void AppendUnsignedShorts(const char * name, unsigned short * values, int valuesRank , int *
shape, const char * dim);
void AppendUnsignedInts(const char * name, unsigned int * values, int valuesRank , int * shape,
const char * dim);
void AppendUnsignedLongLongs(const char * name, unsigned long long * values, int valuesRank , int
* shape, const char * dim);
void AppendFloats(const char * name, float * values, int valuesRank , int * shape, const char *
dim);
void AppendDoubles(const char * name, double * values, int valuesRank , int * shape, const char *
dim);
void AppendStrings(const char * name, char ** values, int valuesRank , int * shape, const char *
dim);
void AppendDateTimes(const char * name, time_t * values, int valuesRank , int * shape, const char
* dim);
void AppendBytes(const char * name, byte * values, int valuesRank , int * shape, const char *
dim);
void AppendSignedBytes(const char * name, sbyte * values, int valuesRank , int * shape, const
char * dim);
void AppendBooleans(const char * name, bool * values, int valuesRank , int * shape, const char *
dim);
```

Parameters

name – variable name

values – values to append. Array “values” is flat (multidimensional array, C-Style stretched to flat array).

valuesRank – rank of values. Values must have rank equal or one less than variable rank

shape – shape of the multidimensional value to append

dim – dimension to append to

If specified name is not found, exception would be thrown.

If specified variable has another type, exception would be thrown.

Attribute method group

Adds scalar attribute to DataSet

```
void PutAttShort(const char * name, short value);
void PutAttInt(const char * name, int value);
void PutAttLongLong(const char * name, long long value);
void PutAttUnsignedShort(const char * name, unsigned short value);
void PutAttUnsignedInt(const char * name, unsigned int value);
void PutAttUnsignedLongLong(const char * name, unsigned long long value);
void PutAttFloat(const char * name, float value);
void PutAttDouble(const char * name, double value);
void PutAttString(const char * name, char * value);
void PutAttDateTime(const char * name, time_t value);
void PutAttByte(const char * name, byte value);
void PutAttSignedByte(const char * name, sbyte value);
void PutAttBoolean(const char * name, bool value);
```

Parameters

name – *attribute* name

value – value of attribute

Returns scalar attribute of DataSet

```
short GetAttShort(const char * name);
int GetAttInt(const char * name);
long long GetAttLongLong(const char * name);
unsigned short GetAttUnsignedShort(const char * name);
unsigned int GetAttUnsignedInt(const char * name);
unsigned long long GetAttUnsignedLongLong(const char * name);
float GetAttFloat(const char * name);
double GetAttDouble(const char * name);
char * GetAttString(const char * name);
time_t GetAttDateTime(const char * name);
byte GetAttByte(const char * name);
sbyte GetAttSignedByte(const char * name);
bool GetAttBoolean(const char * name);
```

Parameters

name – *attribute* name

If specified name is not found, exception would be thrown.

If specified attribute has another type, exception would be thrown.

Adds array attribute to DataSet

```
void PutAttShorts(const char * name, short * values, int length);
void PutAttInts(const char * name, int * values, int length);
void PutAttLongLongs(const char * name, long long * values, int length);
void PutAttUnsignedShorts(const char * name, unsigned short * values, int length);
void PutAttUnsignedInts(const char * name, unsigned int * values, int length);
void PutAttUnsignedLongLongs(const char * name, unsigned long long * values, int length);
void PutAttFloats(const char * name, float * values, int length);
void PutAttDoubles(const char * name, double * values, int length);
void PutAttStrings(const char * name, char ** values, int length);
void PutAttDateTimes(const char * name, time_t * values, int length);
void PutAttBytes(const char * name, byte * values , int length);
void PutAttSignedBytes(const char * name, sbyte * values, int length);
void PutAttBooleans(const char * name, bool * values, int length);
```

name – *attribute* name

values – values of attribute

length – length of values

Returns array attribute from DataSet

```
short * GetAttShorts(const char * name, int * length);
int * GetAttInts(const char * name, int * length);
long long * GetAttLongLongs(const char * name, int * length);
unsigned short * GetAttUnsignedShorts(const char * name, int * length);
unsigned int * GetAttUnsignedInts(const char * name, int * length);
unsigned long long * GetAttUnsignedLongLongs(const char * name, int * length);
float * GetAttFloats(const char * name, int * length);
double * GetAttDoubles(const char * name, int * length);
char ** GetAttStrings(const char * name, int * length);
time_t * GetAttDateTimes(const char * name, int * length);
byte * GetAttBytes(const char * name, int * length);
sbyte * GetAttSignedBytes(const char * name, int * length);
bool * GetAttBooleans(const char * name, int * length);
```

name – *attribute* name

length – length of returned attribute

If specified name is not found, exception would be thrown.

If specified attribute has another type, exception would be thrown.

Metadata method group

```
char ** Dimensions(int * count);
```

Returns array or dataset dimension names.

Parameters

count-total count of dimensions in dataset

```
char ** Variables(int * count);
```

Returns array or dataset variables names.

Parameters

count-total count of variables in dataset

```
char ** Attributes(int * count);
```

Returns array or dataset attributes names.

Parameters

count-total count of attributes in dataset

```
DataType GetVariableType(const char* name);
```

Returns variable type

Parameters

name-variable name

If specified name is not found, exception would be thrown.

```
DataType AttributeType(const char * name);
```

Returns attribute type

Parameters

name-attribute name

If specified name is not found, exception would be thrown.

```
int GetDimensionLength(const char* name);
```

Returns dimension length

Parameters

name-dimension name

If specified name is not found, exception would be thrown.

Other methods

```
CppVariable & Variable(const char * name);
```

Returns variable with specified name

Parameters

name-variable name

If specified name is not found, exception would be thrown.

```
void Commit();
```

Commits changes in dataset

```
ScSData::DataSet^ GetDataSet();
```

Returns pointer to managed DataSet class object

```
void SetAutoCommitEnabled(bool value);
```

Sets AutoCommitEnabled property

value-new property value

Checks if dataset contains variable with specified name

```
bool Contains(const char name[]);
```

name-name to check

CppVariable class methods

Attribute method group

Adds scalar attribute to Variable

```
void PutAttShort(const char * name, short value);
void PutAttInt(const char * name, int value);
void PutAttLongLong(const char * name, long long value);
void PutAttUnsignedShort(const char * name, unsigned short value);
void PutAttUnsignedInt(const char * name, unsigned int value);
void PutAttUnsignedLongLong(const char * name, unsigned long long value);
void PutAttFloat(const char * name, float value);
void PutAttDouble(const char * name, double value);
void PutAttString(const char * name, char * value);
void PutAttDateTime(const char * name, time_t value);
void PutAttByte(const char * name, byte value);
void PutAttSignedByte(const char * name, sbyte value);
void PutAttBoolean(const char * name, bool value);
```

Parameters

name – attribute name

value – value of attribute

Returns scalar attribute of Variable

```
short GetAttShort(const char * name);
int GetAttInt(const char * name);
long long GetAttLongLong(const char * name);
unsigned short GetAttUnsignedShort(const char * name);
```

```
unsigned int GetAttUnsignedInt(const char * name);
unsigned long long GetAttUnsignedLongLong(const char * name);
float GetAttFloat(const char * name);
double GetAttDouble(const char * name);
char * GetAttString(const char * name);
time_t GetAttDateTime(const char * name);
byte GetAttByte(const char * name);
sbyte GetAttSignedByte(const char * name);
bool GetAttBoolean(const char * name);
```

Parameters

name – *attribute* name

If specified name is not found, exception would be thrown.

If specified attribute has another type, exception would be thrown.

Adds array attribute to Variable

```
void PutAttShorts(const char * name, short * values, int length);
void PutAttInts(const char * name, int * values, int length);
void PutAttLongLongs(const char * name, long long * values, int length);
void PutAttUnsignedShorts(const char * name, unsigned short * values, int length);
void PutAttUnsignedInts(const char * name, unsigned int * values, int length);
void PutAttUnsignedLongLongs(const char * name, unsigned long long * values, int length);
void PutAttFloats(const char * name, float * values, int length);
void PutAttDoubles(const char * name, double * values, int length);
void PutAttStrings(const char * name, char ** values, int length);
void PutAttDateTimes(const char * name, time_t * values, int length);
void PutAttBytes(const char * name, byte * values , int length);
void PutAttSignedBytes(const char * name, sbyte * values, int length);
void PutAttBooleans(const char * name, bool * values, int length);
```

Parameters

name – *attribute* name

values – values of attribute.

length – length of values

Returns array attribute from Variable

```
short * GetAttShorts(const char * name, int * length);
int * GetAttInts(const char * name, int * length);
long long * GetAttLongLongs(const char * name, int * length);
unsigned short * GetAttUnsignedShorts(const char * name, int * length);
unsigned int * GetAttUnsignedInts(const char * name, int * length);
unsigned long long * GetAttUnsignedLongLongs(const char * name, int * length);
float * GetAttFloats(const char * name, int * length);
double * GetAttDoubles(const char * name, int * length);
char ** GetAttStrings(const char * name, int * length);
time_t * GetAttDateTimes(const char * name, int * length);
byte * GetAttBytes(const char * name, int * length);
sbyte * GetAttSignedBytes(const char * name, int * length);
bool * GetAttBooleans(const char * name, int * length);
```

Parameters

name – *attribute* name

length – length of returned attribute

If specified name is not found, exception would be thrown.

If specified attribute has another type, exception would be thrown.

Metadata method group

```
char ** Attributes(int * count);
```

Returns array or variable attributes names.

Parameters

count-total count of attributes in variable

```
char ** Dimensions(int * count);
```

Returns array or variable dimensions names.

Parameters

count-total count of dimensions in variable

```
int GetShape(int index)
```

Returns length or of specified variable dimension

Parameters

index-dimension index

```
DataType GetType();
```

Returns variable type.

```
int GetRank();
```

Returns variable rank.

```
DataType GetAttributeType(const char * name);
```

Returns attribute type

Parameters

name-attribute name

If specified name is not found, exception would be thrown.