MetraTech Metering SDK Reference Guide

Version 1.1

January 2000

MetraTech Metering SDK

Description

The MetraTech Metering Software Development Kit.

MTMeter Class

class MTMeter

The MTMeter object controls the rest of the metering library. Each application using the library should have a MTMeter object that is used to generate **MTMeterSession** objects.

Class Members

Public:

MTMeter(MTMeterConfig & config)
Constructor. Init must still be called before using other methods in class.

virtual ~MTMeter()
Destructor. The destructor calls **Shutdown** if **Shutdown** hasn't been called already.

BOOL Startup()
Initialize the object. This function must be called before any other method is used in this class.

BOOL Shutdown()
Shuts down the class. This function frees up any memory used.

MTMeterSession * CreateSession(const char * serviceName)
Generates a new **MTMeterSession** to hold property values that are used to describe a metered event.

unsigned long GetLastError() const
Return the error code, or 0 if there was no error.

MTMeterError * GetLastErrorObject() const
Returns an **MTMeterError** object that holds information about the last error. The MTMeterError object must be deleted after it is used.

static void EnableDiagnosticLogging(MTDebugLogLevel level, FILE * logStream)
Call this function if you're having problems debugging an application using the metering object. The log produced can help track down bugs and can help MetraTech diagnose problems. This method only enables logging in the debug version of the library. In the release version, it does nothing.

Protected:

NetMeterAPI * mpAPI
Object used by the implementation of MTMeter

Private:

ErrorObject * mpErrObj
Object used by the implementation of MTMeter

void SetError(unsigned long aCode, const char * apModule, int aLine, const char * apProcedure)
method used by the implementation of MTMeter

void SetError(const ErrorObject * apError)
method used by the implementation of MTMeter

MTMeter::CreateSession

MTMeterSession * MTMeter::CreateSession(const char * *serviceName***)**

Generates a new **MTMeterSession** to hold property values that are used to describe a metered event.

Return Value Session that will hold properties. Must be deleted when no longer needed.

Parameters *serviceName*
Name of service. Must match the name of a service defined on the metering server.

MTMeter::EnableDiagnosticLogging

static void MTMeter::EnableDiagnosticLogging(MTDebugLogLevel *level***, FILE *** *logStream***)**

Call this function if you're having problems debugging an application using the metering object. The log produced can help track down bugs and can help MetraTech diagnose problems. This method only enables logging in the debug version of the library. In the release version, it does nothing.

Parameters

level
Detail level used to log messages. See **MTDebugLogLevel** for values.

logStream
stdio file pointer where logging messages are sent.

MTMeter::GetLastError

unsigned long MTMeter::GetLastError(void)

Return the error code, or 0 if there was no error.

Return Value

Error code. For Windows NT, the code is either a Win32 error or it's an error code defined in sdk_msg.h.

MTMeter::GetLastErrorObject

MTMeterError * MTMeter::GetLastErrorObject(void)

Returns an **MTMeterError** object that holds information about the last error. The MTMeterError object must be deleted after it is used.

Return Value

MTMeteringError object representing the last error that occurred in the MTMeter object or NULL if there was no error. The object must be deleted after use.

MTMeter::MTMeter

MTMeter::MTMeter(MTMeterConfig & *config***)**

Constructor. Init must still be called before using other methods in class.

Parameters

config
Configuration object which holds specifics about the transport and protocol used to send messages to the server.

See Also

MTMeterConfig

MTMeter::Shutdown

BOOL MTMeter::Shutdown(void)

Shuts down the class. This function frees up any memory used.

Return Value

If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

MTMeter::Startup

BOOL MTMeter::Startup(void)

Initialize the object. This function must be called before any other method is used in this class.

Return Value

If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

MTMeter::~MTMeter

virtual MTMeter::~MTMeter(void)

Destructor. The destructor calls **Shutdown** if **Shutdown** hasn't been called already.

MTMeterConfig Class

class MTMeterConfig

The MTMeterConfig object holds configuration information about the transport and protocol used by the SDK.

Class Members

Public:

virtual ~MTMeterConfig()
Destructor.

enum Protocol
Protocol choices used to send sessions. Currently MSIX is the only protocol supported.

Protected:

virtual NetMeterAPI * GetAPI()
Used by MTMeter only. Do not use this function in any other case.

MTMeterConfig::~MTMeterConfig

virtual MTMeterConfig::~MTMeterConfig(void)

Destructor.

MTMeterError Class

class MTMeterError

The MeteringError object holds an error code and allows you to generate an error message for each code. The MeteringError object also holds the time an error occurred and information of interest to the developer.

Class Members

virtual ~MTMeterError()
Destructor.

virtual unsigned long GetErrorCode() const
Return the error code.

virtual BOOL GetErrorMessage(wchar_t * buffer, int & bufferSize) const
Get the error message in Unicode. GetErrorMessage fills the buffer as far as possible and terminates it with a null character. If the buffer size is zero, GetErrorMessage returns the number of wchar_t values to hold the message and a terminating null character.

virtual BOOL GetErrorMessage(char * buffer, int & bufferSize) const
Get the error message in ASCII. GetErrorMessage fills the buffer as far as possible and terminates it with a null character. If the buffer size is zero, GetErrorMessage returns the number of char values to hold the message and a terminating null character.

virtual int GetErrorMessageEx(char * buffer, int & bufferSize) const
Returns information to the programmer that can be useful in diagnosing the source of an error. GetErrorMessageEx fills the buffer as far as possible and terminates it with a null character. If the buffer size is zero, GetErrorMessage returns the number of char values to hold the message and a terminating null character.

virtual time_t GetErrorTime() const
Returns the time the error occurred.

MTMeterError()
Constructor. The constructor object is protected because MTMeterError objects cannot be created directly.

MTMeterError::GetErrorCode

virtual unsigned long MTMeterError::GetErrorCode(void)

Return the error code.

Return Value

Error code. For Windows NT, the code is either a Win32 error or it's an error code defined in sdk_msg.h.

MTMeterError::GetErrorMessage

virtual BOOL MTMeterError::GetErrorMessage(wchar_t * *buffer***, int &** *bufferSize***)**

Get the error message in Unicode. GetErrorMessage fills the buffer as far as possible and terminates it with a null character. If the buffer size is zero, GetErrorMessage returns the number of wchar_t values to hold the message and a terminating null character.

Return Value FALSE if the buffer is NULL, otherwise TRUE.

Parameters

buffer
Pointer to buffer of type wchar_t that will hold the error message.

bufferSize
Size of buffer (number of wchar_t values it can hold). If zero, it will be set to the size of the buffer required to hold the string and the terminating zero.

MTMeterError::GetErrorMessage

virtual BOOL MTMeterError::GetErrorMessage(char * *buffer***, int &** *bufferSize***)**

Get the error message in ASCII. GetErrorMessage fills the buffer as far as possible and terminates it with a null character. If the buffer size is zero, GetErrorMessage returns the number of char values to hold the message and a terminating null character.

Return Value FALSE if the buffer is NULL, otherwise TRUE.

Parameters

buffer
Pointer to buffer of type char that will hold the error message.

bufferSize
Size of buffer (number of char values it can hold). If zero, it will be set to the size of the buffer required to hold the string and the terminating zero.

MTMeterError::GetErrorMessageEx

virtual int MTMeterError::GetErrorMessageEx(char * *buffer***, int &** *bufferSize***)**

Returns information to the programmer that can be useful in diagnosing the source of an error. GetErrorMessageEx fills the buffer as far as possible and terminates it with a null character. If the buffer size is zero, GetErrorMessage returns the number of char values to hold the message and a terminating null character.

Return Value FALSE if the buffer is NULL, otherwise TRUE.

Parameters

buffer
Pointer to buffer of type char that will hold the message.

bufferSize
Size of buffer (number of char values it can hold). If zero, it will be set to the size of the buffer required to hold the string and the terminating zero.

MTMeterError::GetErrorTime

virtual time_t MTMeterError::GetErrorTime(void)

Returns the time the error occurred.

Return Value time_t value holding the time the error occurred, in GMT.

MTMeterError::MTMeterError

MTMeterError::MTMeterError(void)

Constructor. The constructor object is protected because MTMeterError objects cannot be created directly.

MTMeterError::~MTMeterError

virtual MTMeterError::~MTMeterError(void)

Destructor.

MTMeterHTTPConfig Class

class MTMeterHTTPConfig

The MTMeterConfig object holds configuration information about the HTTP transport and protocol used by the SDK.

MTMeterHTTPConfig(const char * proxyName = NULL, Protocol prot = MSIX_PROTOCOL)
Constructor. You must still call Init to initialize the Metering SDK.

virtual ~MTMeterHTTPConfig()
Destructor. The destructor calls close if the Metering SDK hasn't been closed already.

void AddServer(int priority, const char * serverName, int port, BOOL secure, const char * username, const char * password)
Add a Metering Server to be used to meter sessions. More than one server may be added. The library will attempt to send sessions to the server with the highest priority first, then attempt to send to servers with lower priority if the attempt fails. The higher the priority argument value, the higher the priority of the server. If more than one server has the same same priority, the library will pick any one of these at random.

void SetConnectTimeout(int timeout)
Set the duration in milliseconds that each operation will wait before timing out. If a network operation takes longer than this timeout, the Metering SDK will try again until all its retries have been used. If the SDK is still unable to send the message to the server, the message will be sent to the next server added with **AddServer**. If the message fails to be sent to any server, the operation will return an error.

int GetConnectTimeout() const
Retrieves the timeout value in milliseconds.

void SetConnectRetries(int retries)
Sets the number of retries to make before giving up. If a network operation fails more times than this for any reason, the Metering SDK will attempt to resend the message to the next server listed with **AddServer**.

int GetConnectRetries() const
Get the number of retries each network operation will make before moving to the next server.

Class Members

Protected:

virtual NetMeterAPI * GetAPI()
Called by MTMeter. Do not use this function in other cases.

Protected:

MSIXNetMeterAPI * mpAPI
Object used by the implementation of MTMeterHTTPConfig.

MTMeterHTTPConfig::AddServer

void MTMeterHTTPConfig::AddServer(int *priority***, const char *** *serverName***, int** *port***, BOOL** *secure***, const char *** *username***, const char *** *password***)**

Add a Metering Server to be used to meter sessions. More than one server may be added. The library will attempt to send sessions to the server with the highest priority first, then attempt to send to servers with lower priority if the attempt fails. The higher the priority argument value, the higher the priority of the server. If more than one server has the same same priority, the library will pick any one of these at random.

Parameters

priority
Priority of the server. The server with the highest value as this argument will be used first.

serverName
Hostname of the server.

port
Port number on the server. Usually port 80 when not using SSL and 443 when using SSL. A value from the PortNumbers enumeration can be used for this argument.

secure
If TRUE, use SSL to encrypt all communications. If FALSE, don't use encryption when sending data.

username
Username used for HTTP authentication on the server.

password
Password used for HTTP authentication on the server.

MTMeterHTTPConfig::GetConnectRetries

int MTMeterHTTPConfig::GetConnectRetries(void)

Get the number of retries each network operation will make before moving to the next server.

Return Value Number of retries to attempt

MTMeterHTTPConfig::GetConnectTimeout

int MTMeterHTTPConfig::GetConnectTimeout(void)

Retrieves the timeout value in milliseconds.

Return Value Timeout value in milliseconds.

MTMeterHTTPConfig::MTMeterHTTPConfig

MTMeterHTTPConfig::MTMeterHTTPConfig(const char * proxyName = *NULL*, **Protocol prot =** *MSIX_PROTOCOL*)

Constructor. You must still call Init to initialize the Metering SDK.

Parameters

NULL
An optional proxy server setting. If the proxy server's name is "proxy1" and runs on port 8000, the syntax of this parameter would be "http:://proxy1:8000". If the parameter is NULL, no proxy server will be used.

MSIX_PROTOCOL
Protocol to use when sending messages to the server. Currently only the Metered Session Information Exchange (MSIX) protocol is supported.

MTMeterHTTPConfig::SetConnectRetries

void MTMeterHTTPConfig::SetConnectRetries(int *retries***)**

Sets the number of retries to make before giving up. If a network operation fails more times than this for any reason, the Metering SDK will attempt to resend the message to the next server listed with **AddServer**.

Parameters *retries*
Number of retries to attempt before trying the next host.

MTMeterHTTPConfig::SetConnectTimeout

void MTMeterHTTPConfig::SetConnectTimeout(int *timeout***)**

Set the duration in milliseconds that each operation will wait before timing out. If a network operation takes longer than this timeout, the Metering SDK will try again until all its retries have been used. If the SDK is still unable to send the message to the server, the message will be sent to the next server added with **AddServer**. If the message fails to be sent to any server, the operation will return an error.

Parameters *timeout*
Timeout value in milliseconds.

MTMeterHTTPConfig::~MTMeterHTTPConfig

virtual MTMeterHTTPConfig::~MTMeterHTTPConfig(void)

Destructor. The destructor calls close if the Metering SDK hasn't been closed already.

MTMeterSession Class

class MTMeterSession

The MTMeterSession object holds the property values for a metered session. Objects of this type are created by **MTMeter::CreateSession**. **InitProperty** should be called for each property value before **Save** or **Close** is called to send the properties to the server. **SetProperty** may be called to modify properties before the session is Closed, and **GetProperty** can be used to retrieve property values from the session.

Class Members

MTMeterSession()
Constructor. You cannot construct MTMeterSession objects directly.

virtual ~MTMeterSession()
Destructor. For the properties of the session to be sent to the metering server, **Close** or **Save** must be called before deleting the object.

virtual BOOL Close()
After each property has been initialized to the correct value, Close sends the session and its parents and children, when appropriate, to the metering server. The session is marked complete by a call to Close and further modification is not allowed on the session object. The metering server is allowed to begin processing a session after it has been closed. When a session is closed, any parents of the session are saved but will still be modifiable. Any children of the session will be closed and will not allow further modification.

virtual BOOL Save()
After each property has been initialized to appropriate values, Save sends the session and its parents and children when appropriate to the metering server. After a session has been saved, it is still possible to modify property values by calling **SetProperty**. A session can be saved any number of times before finally being closed. If a session could potentially remain open for a long period of time, saving the session periodically will ensure that the properties are sent to the server. When a session is saved, any parents and children of the session will be saved and will still allow further modification.

virtual void GetSessionID(char * sessionId) const
Get a session identifier that uniquely identifies a session on the Metering Server.

virtual void GetReferenceID(char * referenceId) const
Get a reference ID that can be displayed to a user. This ID is shorter and more easily displayed and read by the user. The reference ID samples part of the session ID. Therefore it doesn't uniquely identify a session. There is a small probability that a user will have two sessions with the same reference ID. Other information about the session can then be used to determine which session a user is referring to.

virtual unsigned long GetLastError() const
Return the error code, or 0 if there was no error.

virtual MTMeterError * GetLastErrorObject() const
Return an **MTMeterError** object that holds information about the last error. The **MTMeterError** object must be deleted after it is used.

virtual BOOL InitProperty(const char * name, const wchar_t * val)
Initialize and initially set a property's Unicode value. InitProperty must be called before **SetProperty** may be called. InitProperty may not be called after **Save** or **Close** has been called.

virtual BOOL InitProperty(const char * name, const char * val)
Initialize and initially set a property's ASCII value. InitProperty must be called before **SetProperty** may be called. InitProperty may not be called after **Save** or **Close** has been called.

virtual BOOL InitProperty(const char * name, int val)
Initialize and initially set a property's integer value. InitProperty must be called before **SetProperty** may be called. InitProperty may not be called after **Save** or **Close** has been called.

virtual BOOL InitProperty(const char * name, float val)
Initialize and initially set a property's float value. InitProperty must be called before **SetProperty** may be called. InitProperty may not be called after **Save** or **Close** has been called.

virtual BOOL InitProperty(const char * name, double val)
Initialize and initially set a property's double value. InitProperty must be called before **SetProperty** may be called. InitProperty may not be called after **Save** or **Close** has been called.

virtual BOOL InitProperty(const char * name, time_t timestamp)
Initialize and initially set a property's timestamp value. InitProperty must be called before **SetProperty** may be called. InitProperty may not be called after **Save** or **Close** has been called.

virtual BOOL SetProperty(const char * name, const wchar_t * val)
Modify a session's Unicode string value. **InitProperty** must be called before SetProperty may be called. SetProperty may not be called after **Close** has been called.

virtual BOOL SetProperty(const char * name, const char * val)
Modify a session's ASCII string value. **InitProperty** must be called before SetProperty may be called. SetProperty may not be called after **Close** has been called.

virtual BOOL SetProperty(const char * name, int val)
Modify a session's integer value. **InitProperty** must be called before SetProperty may be called. SetProperty may not be called after **Close** has been called.

virtual BOOL SetProperty(const char * name, float val)
Modify a session's float value. **InitProperty** must be called before SetProperty may be called. SetProperty may not be called after **Close** has been called.

virtual BOOL SetProperty(const char * name, double val)
Modify a session's double value. **InitProperty** must be called before SetProperty may be called. SetProperty may not be called after **Close** has been called.

virtual BOOL SetProperty(const char * name, time_t val)
Modify a session's timestamp value. **InitProperty** must be called before SetProperty may be called. SetProperty may not be called after **Close** has been called.

virtual BOOL GetProperty(const char * name, const wchar_t * * val)
Retrieve a session's Unicode string value. The property must have been initialized before GetProperty can return it.

virtual BOOL GetProperty(const char * name, const char * * val)
Retrieve a session's ASCII string value. The property must have been initialized before GetProperty can return it.

virtual BOOL GetProperty(const char * name, int & val)
Retrieve a session's integer value. The property must have been initialized before GetProperty can return it.

virtual BOOL GetProperty(const char * name, float & val)
Retrieve a session's float value. The property must have been initialized before GetProperty can return it.

virtual BOOL GetProperty(const char * name, double & val)
Retrieve a session's double value. The property must have been initialized before GetProperty can return it.

virtual BOOL GetProperty(const char * name, time_t & timestamp)
Retrieve a session's time value, in GMT. The property must have been initialized before GetProperty can return it.

virtual MTMeterSession * CreateChildSession(const char * serviceName)
Create a child of this session. Any number of children can be created for a parent session. Once **Close** has been called on a session, no more children can be created. Sessions that have been saved can still have more children added to them. When the child is deleted, it is removed from the parent. When a parent session is deleted, it deletes any children still connected to it.

MTMeterSession::Close

virtual BOOL MTMeterSession::Close(void)

After each property has been initialized to the correct value, Close sends the session and its parents and children, when appropriate, to the metering server. The session is marked complete by a call to Close and further modification is not allowed on the session object. The metering server is allowed to begin processing a session after it has been closed. When a session is closed, any parents of the session are saved but will still be modifiable. Any children of the session will be closed and will not allow further modification.

Return Value If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

MTMeterSession::CreateChildSession

virtual MTMeterSession * MTMeterSession::CreateChildSession(const char * *serviceName*)

Create a child of this session. Any number of children can be created for a parent session. Once **Close** has been called on a session, no more children can be created. Sessions that have been saved can still have more children added to them. When the child is deleted, it is removed from the parent. When a parent session is deleted, it deletes any children still connected to it.

Return Value Child MTMeterSession. Either the child must be deleted, or its parent must be deleted. If CreateChildSession returns NULL, **GetLastErrorObject** can be called to get more information.

Parameters *serviceName*
Service name of child session.

MTMeterSession::GetLastError

virtual unsigned long MTMeterSession::GetLastError(void)

Return the error code, or 0 if there was no error.

Return Value Error code. For Windows NT, the code is either a Win32 error or it's an error code defined in sdk_msg.h.

MTMeterSession::GetLastErrorObject

virtual MTMeterError * MTMeterSession::GetLastErrorObject(void)

Return an **MTMeterError** object that holds information about the last error. The **MTMeterError** object must be deleted after it is used.

Return Value

MTMeterError object representing the last error that occurred in the MTMeterSession object or NULL if there was no error. The object must be deleted after use.

MTMeterSession::GetProperty

virtual BOOL MTMeterSession::GetProperty(const char * *name***, double &** *val***)**

Retrieve a session's double value. The property must have been initialized before GetProperty can return it.

Return Value

If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters

name
Property name. The name must match the property name in the service definition.

val
A reference to a double that will hold the property value.

MTMeterSession::GetProperty

virtual BOOL MTMeterSession::GetProperty(const char * *name***, int &** *val***)**

Retrieve a session's integer value. The property must have been initialized before GetProperty can return it.

Return Value

If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters

name
Property name. The name must match the property name in the service definition.

val
A reference to an integer that will hold the property value.

MTMeterSession::GetProperty

virtual BOOL MTMeterSession::GetProperty(const char * *name*, **float &** *val***)**

Retrieve a session's float value. The property must have been initialized before GetProperty can return it.

Return Value

If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters

name
Property name. The name must match the property name in the service definition.

val
A reference to a float that will hold the property value.

MTMeterSession::GetProperty

virtual BOOL MTMeterSession::GetProperty(const char * *name*, **const wchar_t * *** *val***)**

Retrieve a session's Unicode string value. The property must have been initialized before GetProperty can return it.

Return Value

If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters

name
Property name. The name must match the property name in the service definition.

val
Pointer to a wchar_t pointer that will point to the Unicode string value. The value is constant and must not be modified.

MTMeterSession::GetProperty

virtual BOOL MTMeterSession::GetProperty(const char * *name***, time_t &** *timestamp***)**

Retrieve a session's time value, in GMT. The property must have been initialized before GetProperty can return it.

Return Value

If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters

name
Property name. The name must match the property name in the service definition.

timestamp
A reference to a time_t that will hold the property value, in GMT.

MTMeterSession::GetProperty

virtual BOOL MTMeterSession::GetProperty(const char * *name***, const char * *** *val***)**

Retrieve a session's ASCII string value. The property must have been initialized before GetProperty can return it.

Return Value

If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters

name
Property name. The name must match the property name in the service definition.

val
Pointer to a char pointer that will point to the ASCII string value. The value is constant and must not be modified.

MTMeterSession::GetReferenceID

virtual void MTMeterSession::GetReferenceID(char * *referenceId***)**

Get a reference ID that can be displayed to a user. This ID is shorter and more easily displayed and read by the user. The reference ID samples part of the session ID. Therefore it doesn't uniquely identify a session. There is a small probability that a user will have two sessions with the same reference ID. Other information about the session can then be used to determine which session a user is referring to.

Parameters

referenceId
Pointer to a character buffer at least 10 bytes long. This will hold the reference ID and a null terminating byte. The reference ID is a nine character string made up of uppercase characters and numbers. The size of this buffer is not checked.

MTMeterSession::GetSessionID

virtual void MTMeterSession::GetSessionID(char * *sessionId***)**

Get a session identifier that uniquely identifies a session on the Metering Server.

Parameters

sessionId
Pointer to a character buffer at least 23 bytes long. This will hold the session ID and a null terminating byte. The size of this buffer is not checked.

MTMeterSession::InitProperty

virtual BOOL MTMeterSession::InitProperty(const char * *name***, const char *** *val***)**

Initialize and initially set a property's ASCII value. InitProperty must be called before **SetProperty** may be called. InitProperty may not be called after **Save** or **Close** has been called.

Return Value

If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters

name
Property name. The name must match the property name in the service definition.

val
Pointer to an ASCII string that holds the property's value. The property must be specified as an ASCII string in the service definition.

MTMeterSession::InitProperty

virtual BOOL MTMeterSession::InitProperty(const char * *name***, double** *val***)**

Initialize and initially set a property's double value. InitProperty must be called before **SetProperty** may be called. InitProperty may not be called after **Save** or **Close** has been called.

Return Value

If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters

name
Property name. The name must match the property name in the service definition.

val
The property's double value. The property must be specified as a double in the service definition.

MTMeterSession::InitProperty

virtual BOOL MTMeterSession::InitProperty(const char * *name***, time_t** *timestamp***)**

Initialize and initially set a property's timestamp value. InitProperty must be called before **SetProperty** may be called. InitProperty may not be called after **Save** or **Close** has been called.

Return Value

If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters

name
Property name. The name must match the property name in the service definition.

timestamp
The property's time value, in GMT. The property must be specified as a timestamp in the service definition.

MTMeterSession::InitProperty

virtual BOOL MTMeterSession::InitProperty(const char * *name***, int** *val***)**

Initialize and initially set a property's integer value. InitProperty must be called before **SetProperty** may be called. InitProperty may not be called after **Save** or **Close** has been called.

Return Value
If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters
name
Property name. The name must match the property name in the service definition.

val
The property's integer value. The property must be specified as an integer in the service definition.

MTMeterSession::InitProperty

virtual BOOL MTMeterSession::InitProperty(const char * *name***, float** *val***)**

Initialize and initially set a property's float value. InitProperty must be called before **SetProperty** may be called. InitProperty may not be called after **Save** or **Close** has been called.

Return Value
If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters
name
Property name. The name must match the property name in the service definition.

val
The property's float value. The property must be specified as a float in the service definition.

MTMeterSession::InitProperty

virtual BOOL MTMeterSession::InitProperty(const char * *name***, const wchar_t *** *val***)**

Initialize and initially set a property's Unicode value. InitProperty must be called before **SetProperty** may be called. InitProperty may not be called after **Save** or **Close** has been called.

Return Value

If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters

name
Property name. The name must match the property name in the service definition.

val
Pointer to a Unicode string that holds the property's value. The property must be specified as a Unicode string in the service definition.

MTMeterSession::MTMeterSession

MTMeterSession::MTMeterSession(void)

Constructor. You cannot construct MTMeterSession objects directly.

MTMeterSession::Save

virtual BOOL MTMeterSession::Save(void)

After each property has been initialized to appropriate values, Save sends the session and its parents and children when appropriate to the metering server. After a session has been saved, it is still possible to modify property values by calling **SetProperty**. A session can be saved any number of times before finally being closed. If a session could potentially remain open for a long period of time, saving the session periodically will ensure that the properties are sent to the server. When a session is saved, any parents and children of the session will be saved and will still allow further modification.

Return Value

If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

MTMeterSession::SetProperty

virtual BOOL MTMeterSession::SetProperty(const char * *name***, int** *val***)**

Modify a session's integer value. **InitProperty** must be called before SetProperty may be called. SetProperty may not be called after **Close** has been called.

Return Value

If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters

name
Property name. The name must match the property name in the service definition.

val
The property's integer value. The property must be specified as an integer in the service definition.

MTMeterSession::SetProperty

virtual BOOL MTMeterSession::SetProperty(const char * *name***, const char *** *val***)**

Modify a session's ASCII string value. **InitProperty** must be called before SetProperty may be called. SetProperty may not be called after **Close** has been called.

Return Value

If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters

name
Property name. The name must match the property name in the service definition.

val
The property's ASCII string value. The property must be specified as an ASCII string in the service definition.

MTMeterSession::SetProperty

virtual BOOL MTMeterSession::SetProperty(const char * *name***, time_t** *val***)**

Modify a session's timestamp value. **InitProperty** must be called before SetProperty may be called. SetProperty may not be called after **Close** has been called.

Return Value If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters

name
Property name. The name must match the property name in the service definition.

val
The property's time value, in GMT. The property must be specified as a timestamp in the service definition.

MTMeterSession::SetProperty

virtual BOOL MTMeterSession::SetProperty(const char * *name***, float** *val***)**

Modify a session's float value. **InitProperty** must be called before SetProperty may be called. SetProperty may not be called after **Close** has been called.

Return Value If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters

name
Property name. The name must match the property name in the service definition.

val
The property's float value. The property must be specified as a float in the service definition.

MTMeterSession::SetProperty

virtual BOOL MTMeterSession::SetProperty(const char * *name***, double** *val***)**

Modify a session's double value. **InitProperty** must be called before SetProperty may be called. SetProperty may not be called after **Close** has been called.

Return Value If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters

name
Property name. The name must match the property name in the service definition.

val
The property's double value. The property must be specified as a double in the service definition.

MTMeterSession::SetProperty

virtual BOOL MTMeterSession::SetProperty(const char * *name***, const wchar_t *** *val***)**

Modify a session's Unicode string value. **InitProperty** must be called before SetProperty may be called. SetProperty may not be called after **Close** has been called.

Return Value If TRUE, the function succeeded. If FALSE, the function failed and **GetLastErrorObject** can be called to get more information.

Parameters

name
Property name. The name must match the property name in the service definition.

val
The property's Unicode string value. The property must be specified as a Unicode string in the service definition.

MTMeterSession::~MTMeterSession

virtual MTMeterSession::~MTMeterSession(void)

Destructor. For the properties of the session to be sent to the metering server, **Close** or **Save** must be called before deleting the object.

MTDebugLogLevel

```
enum MTDebugLogLevel {
    MT_LOG_NONE,
    MT_LOG_INFO,
    MT_LOG_DEBUG
};
```

Values that can be passed into MTMeter::EnableLogging

Members

MT_LOG_NONE
Do not display any logging information.

MT_LOG_INFO
Only display informational messages.

MT_LOG_DEBUG
Display informational messages and diagnostic/debugging messages.

MTMeterConfig::Protocol

```
enum MTMeterConfig {
    MSIX_PROTOCOL
};
```

Protocol choices used to send sessions. Currently MSIX is the only protocol supported.

Members

MSIX_PROTOCOL
Metered Session Information Exchange protocol. See www.msix.org for more details.