API Reference for M-Business Anywhere™

Published: January 2008

Part number: DC00302-01-0670-02

Copyright and trademarks

Contents

About this guide

Focus of this guide

This API Reference describes the M-Business Anywhere™ APIs available for use in developing extensions to, and applications for, M-Business Client. This guide covers the following:

- M-Business client extension API — see:
  - "PODS API mechanics" on page 51
  - "PODS object-related and top level objects" on page 67
  - "PODS document-related objects" on page 169
  - "PODS submission-related objects" on page 195
  - "PODS browser-related objects" on page 227
  - "PODS miscellaneous objects" on page 287
  - "PODS event and exception objects" on page 319
- M-Business DOM API — see "PODS DOM-related objects" on page 109
- M-Business XML API — see "M-Business XML API reference" on page 331
- M-Business SOAP API — see "M-Business SOAP API reference" on page 393
- M-Business Date/Time Picker API — see "M-Business Date/Time Picker API" on page 454
- M-Business List Viewer API — see "M-Business List Viewer API" on page 465

All of the API functions and methods are accessible from C; most are also accessible from JavaScript. The content of this guide is oriented to C programming, but the reference material covers API access from both C and JavaScript.

For more information on working with the JavaScript provided in M-Business JavaScript engine, see the following topics in the *Application Developer Guide for M-Business Anywhere*:

- Appendix "M-Business JavaScript engine reference" [*M-Business Anywhere Application Developer Guide*]
- Appendix "M-Business JavaScript engine sample code" [*M-Business Anywhere Application Developer Guide*]

Conventions

Formatting conventions

The following table lists the formatting conventions used throughout this book.

Table 1. Formatting conventions

Item	Treatment	Example
Name of publication	Italic	*Administrator Guide for M-Business Server*
Items on which user is to take an action	Bold	Click the **Reset** button.
Multi-level menu selections	Bold with "»" separator	Choose **Start»Settings» Control Panel**.
Text you type	Bold, fixed width font	Type **`Admin`** in this field.
Text displayed in a file or on the screen	Fixed width font	The screen reads: `Backup Complete`
Keyboard key	Angle brackets	<Enter>
File names and paths	Italic	*.../conf/sync.conf*
Literals in code synopsis	Bold	void **PODSaddRef(**PODSObject* *podsobj***);**
Variables in code synopsis	Italic	void **PODSaddRef(**PODSObject* *podsobj***);**
Variables in text	Angle brackets plus italic	http://<*servername*>:<*port*>

Interface definition language (IDL) organization

Large parts of the M-Business Anywhere client extension API (PODS) and the M-Business XML API are accessible from both JavaScript and C. To make it easier to locate specific information within an interface, features are consistently ordered by IDL keyword. This presents no problem for JavaScript programmers, but when C programmers look up reference material, they must omit the `get` or `set` prefix to the macro name that they must call to access the attribute.

The M-Business Anywhere documentation set

In addition to this document, there are several other iAnywhere Solutions publications available that you may find useful in setting up and using M-Business Server.

Note
Unless otherwise noted, all of these publications are available from: http://www.ianywhere.com/developer/product_manuals/mbusiness_anywhere/
In order for links between different PDF files to work correctly, you must open the files directly from the web site, or download them from the web site into the same local directory.

- *Developer Quick Start Guide for M-Business Anywhere*
- *Release Notes for M-Business Anywhere*
- *User Guide for M-Business Anywhere Client*
- *M-Business Anywhere, an Introduction*
- *Application Developer Guide for M-Business Anywhere*
- *API Reference for M-Business Anywhere*
- *Ensuring Mobile Security from the Device to the Datacenter,* available from http://www.ianywhere.com/whitepapers/ensuring_security.html

Related publications

UltraLite for M-Business Anywhere

For more information about using UltraLite® for M-Business Anywhere, please refer to the following iAnywhere Solutions™ publications:

- *UltraLite for M-Business Anywhere Quick Start*, available from http://www.ianywhere.com/developer/product_manuals/sqlanywhere/1001/en/html/ulagen10/ag-preparing-evb-development.html
- *Exploring the CustDB Samples for UltraLite* http://www.ianywhere.com/developer/product_manuals/sqlanywhere/1001/en/html/ulfoen10/fo-fo-custdb.html
- *UltraLite M-Business Anywhere Programming, version 10*, available from http://www.ianywhere.com/developer/product_manuals/sqlanywhere/1000/en/pdf/ulagen10.pdf

Adaptive Server Anywhere

For more information about Adaptive Server® Anywhere, please refer to the following iAnywhere Solutions publications.

- *SQL Anywhere 10* documentation, available from http://www.ianywhere.com/developer/product_manuals/sqlanywhere/1001/en/html/index.html
- *Setting Up Adaptive Server Anywhere as a Cluster Database Service*, available from http://www.ianywhere.com/developer/technotes/asa_cluster_db_service.html

Recommended references

This guide covers only use of the M-Business Anywhere APIs, via C and JavaScript programming. For a list of recommended references on other development environments that you can use to develop applications for M-Business Client, see "Recommended references" [*M-Business Anywhere Application Developer Guide*].

Contacting iAnywhere Solutions

Technical support

If you need assistance using iAnywhere software, in North America, please contact iAnywhere Technical Support by calling 1-800-8SYBASE (800-879-2273) and then selecting option 3. You can call Monday through Friday (except major US holidays) between 9:00 a.m. and 9:00 p.m. Eastern time. Services will be provided in accordance with your support agreement.

Outside of North America, for your local support number and hours, please see: http://www.sybase.com/contactus/support

Registering as a Named Contact

Calling the 800-number during business hours should always work to get you technical support — a Customer Number is created for you as soon as your purchase is completed. You will find it faster and easier to get technical support, by phone or online, if you have registered as a Named Contact.

When you purchase an iAnywhere product, a *Sybase Technical Support Contact Form* will automatically be emailed to you within 7-10 days. If your company should need to add another Named Contact, or change the one initially registered, call the Technical Support 800-number and request a *Sybase Technical Support Contact Change Form*.

The *Sybase Technical Support Contact Form* will contain your Customer Number, with spaces for you to provide an email address and other identifying information for the Named Contact for your product. Fill in the requested information and fax the form back to the phone number indicated.

When your fax is received, an email will be sent to you, providing your Technical Contact ID number. You can then use this number to speed up the process when you call for technical support, and to access technical support online.

Using the Sybase Online Support Services

A major benefit of using the Sybase Online Support Services is 24x7 availability. Online support also allows you to look up and review past and current support issues.

When you register as a Named Contact, the email sent to you with your Technical Contact ID number also contains instructions for registering and using the Sybase Online Support Services. Follow these instructions to register as a first-time user, or to update your account with information for the new product you have purchased.

If you have any trouble registering for the Sybase Online Support Services, you can of course call iAnywhere Technical Support for assistance!

Application development — customizing iAnywhere software

If you need help with customizing iAnywhere software to better serve your enterprise, please contact iAnywhere Solutions Professional Services at contact_us@ianywhere.com.

Product information

If you need information about other iAnywhere products for your enterprise, please contact iAnywhere Workforce Sales at contact_us@ianywhere.com.

Feedback on documentation

If you have questions or suggestions about this document or other iAnywhere technical publications, please contact iAnywhere Technical Publications at iasdoc@ianywhere.com.

We would like to receive your opinions, suggestions, and feedback on this documentation. Although we do not reply to individual emails, we read all suggestions with interest and attempt to incorporate them in future releases.

Part I. Concepts and Development Guidelines

CHAPTER 1

Introduction

Contents

What is M-Business JavaScript engine?

JavaScript versus PODS

JavaScript provides you, the web developer, a quick and simple language to use for enhancing web pages and servers. A segment of JavaScript functionality is embedded as a small program within a web page which is in turn interpreted and executed by the web client. JavaScript functions that can be called from within a web document are often executed by mouse functions, buttons, or other user-initiated actions.

PODS, on the other hand, provides you with better performance, the capability to perform file operations, interaction with native programs, and increased programming flexibility.

Note
PODS is not currently supported on Symbian OS. This includes the XML database POD, the List Viewer POD, the Date/Time Picker POD, and custom PODS. You may use JavaScript to make any API calls that are available to JavaScript on Symbian OS devices.

Brief description of JavaScript engine

JavaScript is a compact, cross-platform, object-based scripting language that extends the capabilities of HTML. JavaScript is integrated with HTML to allow developers to create interactive web pages. For example, you can create a JavaScript program to pre-validate a form before sending it back to the server, set options based on user preferences, update text displayed in a form's text box, etc. Because JavaScript is downloaded with the HTML page, its execution requires no further interaction with the server.

JavaScript is easy to learn, yet powerful enough for sophisticated scripting tasks. It uses syntax similar to C and C++ and has object-oriented features that use prototype-based inheritance.

M-Business JavaScript engine is iAnywhere Solutions, Inc.' implementation of client-side JavaScript. Many JavaScript features that are not considered high priority for handheld devices are not supported in order to conserve mobile device resources. At the same time, most of the features of PODS are directly available to M-Business JavaScript engine as if they were provided by JavaScript native objects. For details on what M-Business JavaScript engine omits from JavaScript and what it adds from PODS, see the *Application Developer Guide for M-Business Anywhere*.

Specific features of M-Business JavaScript engine

M-Business JavaScript engine enables developers to:

- Dynamically generate HTML pages
- Dynamically change the contents of HTML pages via DHTML
- Dynamically change the contents of forms on HTML pages

- Call a browser to perform various tasks, such as manipulating form fields or form submissions
- Allow users to interact with HTML pages when disconnected, if the page is cached on the device

> **Note**
> Some of the M-Business JavaScript engine functionality results from direct access to PODS objects. There is considerable overlap in the functionality that is available in M-Business JavaScript engine and in PODS. For a comparison of what you can do in each programming environment, see "Choosing a language" on page 9.

What is PODS?

Brief description of PODS

M-Business client extension API, referred to as PODS (Portable Object Delivery System), is a collection of shared object-oriented code libraries on a mobile device that can be used by one or more channels. A POD is a shared library that extends M-Business Client applications running on handheld devices.

- On Palm Computing devices, a POD is a Palm OS shared library with the same limitations as all Palm OS shared libraries. For detailed information on building a POD for a device that uses the Palm OS, see "Building PODS for Palm OS devices" on page 23.
- On Microsoft OS devices, a POD is a DLL. For detailed information on building a POD for a device that uses a Microsoft OS, see "Building PODS for Microsoft OS devices" on page 25.
- On EPOC devices, a POD is a DLL. For detailed information on building a POD for a device that uses EPOC, see "Implementing the PODSPodNew( ) function" on page 27.

You can use PODS in addition to, or instead of, JavaScript in your HTML interface. PODS allows M-Business Anywhere client to perform on-device logic and generate dynamic HTML pages in response to user requests. For example, suppose you have an HTML form that uses JavaScript to check the validity of the data a user enters before submitting the form. You can perform this same validity checking using PODS, so that when a user submits a form during a synchronization, your web application knows that the data it receives is valid. You can also use PODS if you want users to be able to interact with your application in disconnected mode.

> **Note**
> PODS is not currently supported on Symbian OS. This includes the XML database POD, the List Viewer POD, the Date/Time Picker POD, and custom PODS. You may use JavaScript to make any API calls that are available to JavaScript on Symbian OS devices.

Specific features of PODS

PODS enables developers to:

- Export objects to JavaScript
- Dynamically generate HTML pages
- Dynamically change the contents of HTML pages via DHTML
- Dynamically change the contents of forms on HTML pages
- Call a browser to perform various tasks, such as manipulating form fields or form submissions
- Provide on-device logic in native code

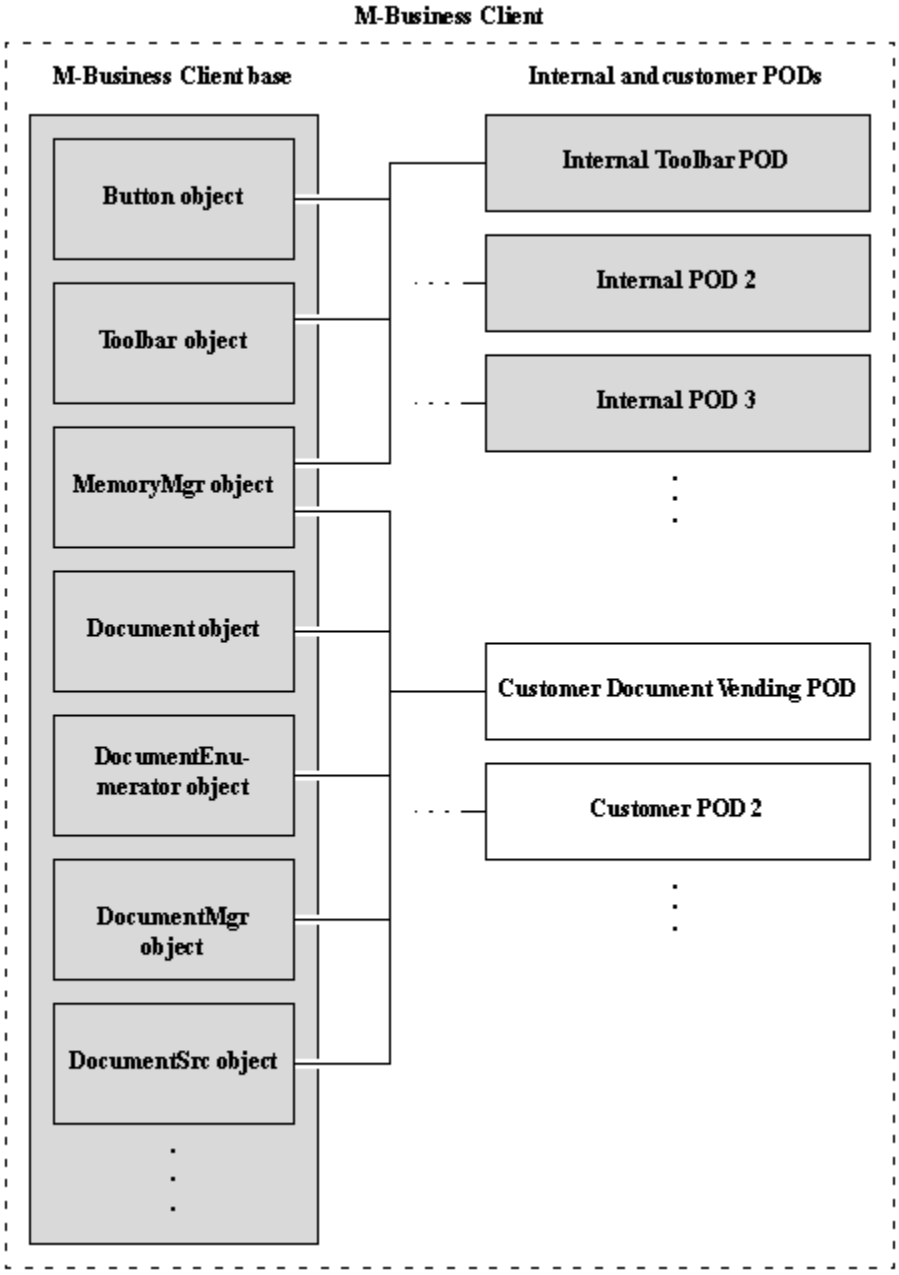
- Control device-specific hardware via vendor-provided APIs (e.g., pagers, scanners, IR)
- Integrate with third-party on-device software via vendor-provided APIs (e.g., databases, MP3 players)
- Allow users to interact with HTML pages when disconnected, if the page is cached on the device
- Handle system events

> **Note**
> Much of the PODS functionality listed above also is accessible from M-Business JavaScript engine. For a comparison of what you can do in each programming environment, see "Choosing a language" on page 9.

PODS in M-Business Client architecture

All the functions available through M-Business Client are provided through internal PODs that use the PODS objects. The M-Business Anywhere client extension API is just the set of internal objects that are exposed for customers to use in creating PODs that extend the basic functionality of M-Business Client. "Figure 2-1" on page 8 illustrates the relationship between the internal objects, internal PODs, and customer created PODs. The customer PODs plug into the system just like the internal PODs.

Figure 2-1



Getting started

Before you can start coding, you have to decide what mix of programming languages to use: JavaScript only, C only, or some mix of the two. Then you must set up your development environment accordingly.

Choosing a language

PODS is written in C and currently has supported bindings only for C. In addition, most of the methods and attributes available through PODS objects are exported to M-Business JavaScript engine. Thus it may be possible to use either JavaScript or C to implement your application. The table below summarizes the considerations involved in choosing the language you will use to develop your M-Business Anywhere client application. In many cases, the best solution will be to use M-Business JavaScript engine for the bulk of the application code, and write C code for PODS objects for use in JavaScript when that is required to implement particular functions.

Table 1. Considerations when choosing a development language

Consideration	JavaScript	C
Portability	Fully cross-platform, including testing and use in desktop browsers	Somewhat platform-specific; although PODS code can be written to be platform neutral, at a minimum the source must be recompiled for each platform
Access to PODS objects	Most	All
Access to OS calls	No	Yes
Execution speed	Generally slower	Generally faster
Memory requirement	Generally higher	Generally lower
Management of hardware	Symbol Technologies scanner only	Any device for which you can obtain or write a driver
Form submission	Yes	Yes
Dynamic generation of HTML pages	Yes	Yes
Dynamically changing contents of HTML forms	Yes	Yes
Maintaining state between pages	Yes	Yes

Consideration	JavaScript	C
Integrating with third party on-device software via APIs	No	Yes

PODS has been supported in M-Business Client by the M-Business Anywhere implementation of JavaScript. JavaScript can call PODS code to extend JavaScript with capabilities not present in the JavaScript language.

PODS can be used to develop a variety of user applications. Many of these applications also could be developed using JavaScript. However, there may still be reasons for using PODS, such as smaller executable code, faster performance, and the ability to make calls directly into the operating system.

Your choice may be completely determined by one or two of the trade-offs in the table above. If you need your application to make OS calls, or perform intensive number crunching as fast as possible, you must choose C, at least to perform those functions.

If you need your application to be fully cross-platform, you probably should choose M-Business JavaScript engine. JavaScript's simple application delivery model and ease of programming make it a great choice for most applications.

If you do not require something that you can get only with C, use JavaScript. It is simpler to program and easier to maintain, and your pages can be tested and used on desktop browsers as well.

Development tools you will need

M-Business JavaScript engine

For M-Business JavaScript engine, the only requirement is a text or HTML editor. You may use whatever editor you like. You may also want to take advantage of one of the desktop debugging tools that are available for JavaScript.

M-Business XML conduit and M-Business XML API

In addition to M-Business Server, Application Edition, you will need to have a back-end database in which to store the data you want to mobilize and an application server capable of receiving and formatting HTTP requests.

PODS/C on Palm OS

For Palm OS you need CodeWarrior for Palm OS, version 6.0 or higher, with support for Palm OS 3.0 or higher.

PODS/C on Microsoft OS devices

For Microsoft OS devices you need eMbedded Visual C++ 3.0 and its SDKs.

Example PODS files

A number of sample application files are provided. You will probably want to look these over before you start coding your own application, and perhaps use them as a starting point for writing your code. For

instructions on accessing the sample application files, see "Downloading and working with the PODS sample files" on page 488.

CHAPTER 2

Writing C Code for PODS

Contents

Overview of tasks to create PODS

The tasks you must perform to create PODS with M-Business client extension API (PODS) are summarized below. Each of these tasks is detailed in the sections that follow.

- Before you start writing code to implement your POD, familiarize yourself with the programming considerations that will be involved. See "Understanding the programming considerations" on page 15.
- Determine what components your POD will provide to extend M-Business Client. See "Determining what PODS components your PODS will need" on page 22.
- Implement a `PODSPodNew()` function that will initialize your POD, allocating memory and performing any other initialization functions that you need, and registering any object source documents or other objects that the POD supplies. See "Implementing the PODSPodNew( ) function" on page 27.
- Write the code for your POD's main functionality. In general, this involves writing code that implements a feature and then registering that code with the appropriate PODS manager. See "Writing the code for your PODS' main application functionality" on page 29.
- As necessary, handle the platform-specific issues. See "Handling platform-specific issues" on page 23.
- Install your POD on mobile devices. "Installing PODS on users' devices" on page 33.

Understanding the programming considerations

To fully understand the PODS object model, you should be proficient in C and be comfortable with concepts such as pointers to structures and pointers to functions. Experience in an object-oriented language also is very helpful. If you are not very experienced in C, you still should be able to write simple PODS applications if you have a minimal understanding of the PODS object model.

PODS defines conventions for representing objects, interfaces, and methods. These conventions are more involved than object-oriented languages, where these notions are built in. PODS applications are written in C or C++, with all interfaces C-compatible, to maximize portability.

Both M-Business Client and third parties can define PODS objects. M-Business Client defines browser objects that represent browser entities such as documents, forms, text areas, and form submissions. Each POD defines an object that implements the top-level functionality of the POD. PODs also may define additional PODS objects to export to M-Business JavaScript engine.

Objects, interfaces, and methods

Objects, interfaces, and methods are fundamental notions in PODS. In PODS, an object is an implementation of an interface. Each interface defines a set of methods that can be invoked on the objects that implement the interface.

PODS defines a number of built-in interfaces, each in its own header file or in a header file that defines several closely related interfaces. M-Business Client provides browser objects that implement these interfaces. For example, *podsprefs.h* defines the interface `PODSPrefs`, which includes the following methods:

```
PODSString (*m_getStringValueForKey)
(PODSPrefs*, PODSString);

void (*m_setStringValueForKey)
(PODSPrefs*, PODSString, PODSString);

PODSBoolean (*m_getBoolValueForKey)
(PODSPrefs*, PODSString);

void (*m_setBoolValueForKey)
(PODSPrefs*, PODSString, PODSBoolean);

PODSUInt32 (*m_getUInt32ValueForKey)
(PODSPrefs*, PODSString);

void (*m_setUInt32ValueForKey)
(PODSPrefs*,PODSString,    PODSUInt32);

PODSInt32 (*m_getInt32ValueForKey)
(PODSPrefs*, PODSString);

void (*m_setInt32ValueForKey)
(PODSPrefs*, PODSString,   PODSInt32);

PODSUInt8* (*m_getBytesForKey)
(PODSPrefs*, PODSString,   PODSInt32*);
```



```
void (*m_setBytesForKey)
(PODSPrefs*, PODSString,   PODSUInt8*,    PODSUInt32);
```

Each of the above method definitions show the method's arguments and return type. Note that these are not C function prototypes; they are actually defined in *podsprefs.h* using a more complicated syntax (described below). You invoke these methods via macros.

Each PODS interface has a corresponding C structure to define the objects that implement the interface. For example, structures in *podsprefs.h* define the `PODSPrefs` interface. The *podsprefs.h* header defines a macro for each method in each interface that the *podsprefs.h* header defines. You call the methods via the macros. The first argument to the macro must be a pointer to the object on which you want to invoke the method:

```
PODSPrefs* p;
PODSBoolean b = PODSgetBoolValueForKey(p, PREFS_SHOWIMAGES);
```

Note
M-Business Client does not call the `PODSPod` object's "destroy( )" on page 69 method on loaded PODs until most other objects in the system are destroyed. This means that some PODs may not be able to do required cleanup before they are unloaded.

PODS for C++ programmers

The objective of this section is to leverage your existing knowledge of C++ to help you get familiar with the PODS C environment. If you are not a C++ programmer, this section may teach you a little about C++, but it probably will not help much in familiarizing you with the PODS C environment. You may safely skip this topic and go on to "Implementing a PODS interface in C" on page 19.

At first glance, a C++ programmer might not recognize the features of an OO development system as PODS provides them. Because PODS is compatible with pure ANSI C, none of the syntactic constructions familiar to a C++ programmer are available, but the actual functionality behind them is. You must explicitly tell the compiler what to do at a low level rather than using advanced syntax to hide the advanced features. PODS provides some convenient macros to help when dealing with standard classes. (If you've never investigated how a C++ compiler provides inheritance, virtual method calls, and polymorphism, you may find exploration of the PODS macros illuminating, but this document will focus on syntactic issues.)

The most obvious differences between PODS C source code and C++ source code are the syntax of a virtual method call and the lack of implied upcasting. The pattern is fairly simple but will require some changes of habit. Where in C++ you would write

```
// might be explicit with
// static_cast<ParentClass*>(ChildClassPtr)
ParentClass *foo = ChildClassPtr;
foo->Bar(1, 2);
```

In PODS and C you would write

```
// cast is required
ParentClass *foo = (ParentClass*) ChildClassPtr;
PODSbar(foo, 1, 2);
```


Virtual method calls and member data: vtables

The key to PODS is the object's virtual function table. This is the mechanism by which a virtual method call is resolved to an actual function invocation. A vtable (some authors prefer vtbl) is simply a list of all the virtual methods on an object, organized in such a way that a call expecting to see a parent class' vtable will work without changes on a child class' vtable. In C++, simply declaring a class as inheriting from another class tells the compiler to do this work. In C, you have to write the code explicitly.

For a concrete example, suppose we have the following C++ declarations:

```
class Base
{
protected:
     int               x, y;
public:
     Base(int xx, int yy);
     virtual ~Base();

     int               GetX();
     int               GetY();
     virtual int       Sum();
     virtual float     GetMean();
     virtual void      Set(int xx, int yy);
};

class Child : public Base
{
protected:
     float             mean;
public:
     Child(int xx, int yy);
     virtual ~Child();

     virtual float     GetMean();
     virtual void      Set(int xx, int yy);
     virtual float     Schematize();
}
```

The corresponding declarations in C, as done by PODS, would be as shown below.

Note
Except for `PODSPod`, no PODS object type has any data members, so the parent data `struct` type is not declared for them in the real header files.

```
typedef struct Base Base; // forward
struct BaseVTable
{
    void  (*Destroy)(Base *self); // see next
                                  // section
    int   (*Sum)(Base *self);
    float (*GetMean)(Base *self);
    void  (*Set)(Base *self, int xx, int yy);
};
struct BaseData
{
    int               x, y;
};
struct Base
```



```
{
    struct BaseVTable  *vtable;
    struct BaseData    basedata;
};

extern Base *BaseNew(int xx, int yy); // see next
                                      // section
extern int BaseGetX(Base *self);
extern int BaseGetY(Base *self);

#define PODSsum(b)      (*(b->vtable->Sum))(b)
#define PODSgetMean(b)  (*(b->vtable->GetMean))(b)
#define PODSset(b,xx,yy)(*(b->vtable->Set))(b,xx,yy)

typedef struct Child Child;         // forward
struct ChildVTable
{
    // recapitulate BaseVTable
    void   (*Destroy)(Child *self); // see next
                                    // section
    int    (*Sum)(Child *self);
    float  (*GetMean)(Child *self);
    void   (*Set)(Child *self, int xx, int yy);
    // add new methods
    void   (*Schematize)(Child *self);
};
struct ChildData
{
    float  mean;
};
struct Child
{
    struct ChildVTable  vtable;
    struct BaseData     basedata;
    struct ChildData    childdata;
};
extern Child *ChildNew(int xx, int yy); // see
                                        // next section

#define PODSschematize(c)(*(c->vtable->Schematize))(c)
```

Construction and destruction

Another issue is the construction and destruction of objects. C does not have the operators `new` and `delete`. C has only `malloc()`, `free()`, and the related routines, so explicit routines for creating and initializing objects must be provided. Memory for the vtable of an object must be allocated and filled in correctly. The vtable memory must only be allocated once, no matter how many instances of the object get created. Free vtable memory only once, after all instances of the class are gone. Do this by allocating the vtables for all the classes in a POD as data for the `PODSPod` object itself, because only one instance of that object will ever exist and it will be destroyed last.

Although the complex problems of construction and destruction could occupy several pages, the PODS system is designed such that the base classes for objects a POD author will implement, do not require special construction or destruction. In fact, except for PODSPod, the base classes have no data members at all, and PODSPod has only a copy of a system-provided object pointer which does not need destruction in your code. So your only requirement is that you implement the destroy( ) method (inherited from PODSObject) in the proper way to undo what your <*objectname*> `New` did.

Your construction code must explicitly set up the vtable in your object. In general, there are only two places where a PODS object could be created in your POD: in PODSPodNew() and in methods of other objects, such as a PODSDocumentSrc or a PODSObjectSrc. In both cases, the techniques are basically boiler-plate copies of stereotypes, and are amply demonstrated in the PODS sample code. See Appendix "PODS code samples" on page 487.

Advanced C++ concepts

Many of the more advanced concepts and keywords of C++ have no real equivalent in C object code. In particular, scope controls (such as protected members and private inheritance) and `const` methods exist mainly to tell the C++ compiler how things are supposed to be used; in the PODS C environment, you must enforce such rules yourself.

The closest equivalent to a namespace is a careful partitioning of header files. Features such as function overloading (including default parameters), templates, and operator overloading, are not available in C. In particular, methods with the same name but different signatures, on different objects, can cause problems that might not be caught because of all the casting. This problem is much worse when IDL-generated macros are involved, because the result is conflicting macro definitions.

The `PODSExceptionMgr` provides a very basic level of exception handling, much closer to Windows Structured Exception Handling or the C exception handling implemented by compilers like Borland C for 16-bit Windows. This mechanism is primarily provided as a way to interface with JavaScript exception handling, not as a replacement for C++ exceptions.

Caution
Because C does not provide destructors, you must explicitly control the proper destruction of objects in try/catch/finally blocks, specifically guaranteeing that no memory will be leaked in any code path.

Implementing a PODS interface in C

Now that you know how an interface is defined in C, let us see how to implement a hypothetical `PODSCounter` object: a PODS object that implements the `PODSCounter` interface. One possible implementation would be:

```
typedef struct MyPODSCounter {
  PODSCounter base;
  PODSInt32 count;
} MyPODSCounter;

static void MySetCount(PODSCounter* c, PODSInt32 newCount)
{
  MyPODSCounter* m = (MyPODSCounter* ) c;
  m->count = newCount;
}

static PODSInt32 MyGetCount(PODSCounter* c)
{
  MyPODSCounter* m = (MyPODSCounter* ) c;
  return m->count;
}
```



```
static void MyIncrement(PODSCounter* c)
{
  MyPODSCounter* m = (MyPODSCounter* ) c;
  ++m->count;
}

PODSCounter* MyCounterNew()
{
  static PODSCounterVTable* vtable;
  MyPODSCounter* m;

  if (!vtable) {
    vtable = (PODSCounterVTable* ) malloc(sizeof(PODSCounterVTable));
    vtable->m_setCount = &MySetCount;
    vtable->m_getCount = &MyGetCount;
    vtable->m_increment = &MyIncrement;
  }

  m = (MyPODSCounter* ) malloc(sizeof(MyPODSCounter));
  m->base.vtable = vtable;
  m->count = 0;
  return &m->base;
}
```

In this example, the `MyPODSCounter` structure is used to store objects that implement the `PODSCounter` interface. The `MyPODSCounter` structure extends the `PODSCounter` structure by including a `PODSCounter` as its first element.

`MyPODSCounter` adds a field `i` to hold the actual count in this particular `PODSCounter` implementation. The field `i` is private to the `MyPODSCounter` implementation: it is not visible to code that uses the `PODSCounter` structure.

The above methods `MySetCount`, `MyGetCount`, and `MyIncrement` implement each of the methods defined by the `PODSCounter` interface. Each of these methods takes a `PODSCounter`, not a `MyPODSCounter`, as its first argument, so that each method's definition matches its method definition in the `PODSCounterVTable` structure. Each of these methods can only be called with a `MyPODSCounter` object, so each method uses a typecast to convert the method's first argument to a `MyPODSCounter`. This is a common design pattern in code used to implement a PODS interface.

The method `MyCounterNew` is used to create a new `MyPODSCounter` object that implements the `PODSCounter` interface. The `MyCounterNew` method returns a `PODSCounter`, not a `MyPODSCounter`. The structure `MyPODSCounter` is private to the implementation and callers would use a `MyPODSCounter` object only by accessing its methods through the `PODSCounter` interface. The static variable `vtable` holds a vtable that is shared by all `MyPODSCounter` objects. Vtables should be shared among objects of the same class.

In this example, you could create a new vtable for each `MyPODSCounter` object, but that would only waste space, because all `MyPODSCounter` vtables would contain the same data and never would change.

Note
You cannot use a static variable in a POD on devices running Palm OS, where PODS cannot have any global or static data. For more information, see "Coding without globals on Palm OS" on page 24.

The last line of the method `newMyCounter()` converts a `MyPODSCounter*` into a `PODSCounter*` by taking the address of the `PODSCounter` structure, which is the first element of `MyPODSCounter`. Alternatively, you could write:

```
return (PODSCounter* ) m;
```

Lifetime of method arguments and return values

PODS defines the following policies for the lifetime of arguments to PODS methods and of values returned from methods. Methods implemented by M-Business Client's browser objects follow these policies, so you should follow these policies whenever you call M-Business Client browser methods. In addition, any methods you define in a POD must follow these policies. M-Business Client follows these policies whenever it calls your methods, just as M-Business JavaScript engine code calls a method of an object you have implemented.

- Any string passed to a PODS method is guaranteed to remain valid only for the method call. If a method implementation wants to keep a pointer to the string's characters, that method must create its own private copy of the string.
- Code that calls a PODS method and receives an object or string in return is not responsible for freeing the returned object or string.
- Any string returned by a PODS method is guaranteed to remain valid only until the next call to any PODS method.

  A caller that wants to keep a pointer to a returned string's characters must immediately copy the characters upon return from the method. Note that for PODS objects defined by M-Business Client, returned strings actually are guaranteed to remain valid somewhat longer, until the current event handler has finished execution.
- Any object returned by a PODS method is guaranteed to remain valid for as long as the M-Business Client browser remains on the current page, but you may release it and free up the memory earlier. See "Reference counting: registering and freeing objects" on page 40.

As a PODS author, you must make sure that all strings and objects returned by your POD follow these rules. In particular, be sure that all dynamically allocated memory is freed. The best way to ensure that dynamic memory is freed is to use `PODSMemoryMgr` methods to register strings. Any string registered by `stringRegister()` is automatically freed when the current event handler is finished. See "stringRegister( )" on page 98.

Managing memory

In order to maximize performance on the device, M-Business Client performs major memory clean-up — "garbage collection" functions — only when the user closes M-Business Client. To avoid running out of memory, you must ensure that any allocated memory is freed up as soon as it is no longer needed.

In addition to the preceding topic, "Lifetime of method arguments and return values" on page 21, see the following topics in this guide for details on C programming practices that keep applications from running out of memory on M-Business Client:

- "Construction and destruction" on page 18
- "Registering strings and objects to be freed" on page 40
- "PODSMemoryMgr object" on page 96, especially the "stringRegister( )" on page 98 method
- "Coding without globals on Palm OS" on page 24
- "Passing array data between PODS methods and M-Business JavaScript engine" on page 41

For information on JavaScript programming practices that keep applications from running out of memory on M-Business Client, see "Garbage collection and memory management in M-Business Client" [*M-Business Anywhere Application Developer Guide*].

Tip
If a user never closes M-Business Client on a device, eventually all available memory will be used up. To avoid this potential problem, you should design applications in such a way that users are encouraged to close M-Business Client at the end of an application session.

Determining what PODS components your PODS will need

Before you begin writing code to implement your POD, it is a good idea to look over the PODS objects that are available and determine which ones your POD will need to use. For example, will it create whole pages (`PODSDocumentSrc`) or vend JavaScriptable objects (`PODSObjectSrc`)? How will the POD integrate with the rest of your mobile application?

Handling platform-specific issues

PODS strives to make it as easy as possible to build a POD in the same cross-platform way as M-Business Client itself is built, expanding on the technique used for years on Palm. Typically you write the actual code of your POD as pure portable code, then link in a platform-specific source file. The platform-specific source file is named *podsstartup*<*platform*>*.c*, where <*platform*> is replaced by `palm`, `win`, or `epoc` (and for EPOC, the extension is *.cpp* instead of *.c*).

If your code, including `PODSPodNew()` and the actual implementation of your POD's objects, follows all the rules for all the platforms, and uses no platform-specific constructs, it should work on all platforms without any modification other than changing the following linked-in file: *podsstartup*<*platform*>*.c*. The resulting implementation will then be the correct one for the target platform. See "Implementing the PODSPodNew( ) function" on page 27 and "Implementing the PODSObject object" on page 28.

Different platforms have different rules that you must follow and different issues that you must deal with to ensure that your POD works properly on the target device. These are described in the following sections. The most critical platform-specific issues are summarized in "Handling platform-specific issues" on page 23.

Table 1. Summary of critical platform-specific issues

Platform	Critical issues
Palm OS devices	♦ No writable global variables ♦ Extremely limited heap memory; use storage memory whenever possible ♦ No support for double-byte character sets. See "Building PODS for Palm OS devices" on page 23.
Microsoft OS devices	♦ Supports double-byte character sets

Building PODS for Palm OS devices

On the Palm operating system, you implement a POD as a Palm OS system library. Thus, the POD is dynamically linked with the M-Business Client application, rather than statically linked. How you create a system library project depends on the tools you are using.

- For Metrowerks CodeWarrior IDE, go to your project's **Settings** panel. In the **68K Target** panel's **Project Type** list, choose **Library**, with **Palm OS Library** as the option.
- For GNU tools, compile with the **-shared** option.

In either case, make sure you do the following:

- Link with the `podsstartuppalm.o` object module and make sure this module is first in the link list, or include `podsstartuppalm.c` as the first file in your project.
- Specify any creator ID name you like and a type of `"pods"`.

> **Note**
> If you want your POD to be deleted when M-Business Client is deleted on the Palm, the creator ID must be `"AvGo"`. If it is any other value, the POD remains on the Palm even if M-Business Client is removed. If a POD has a creator ID other than `"AvGo"`, the POD can be removed individually from the Palm using **Delete** in the Applications list.

The PODS distribution provides some sample files that you can copy and edit to make sure that you have correctly set up your project. If you are using Metrowerks CodeWarrior, copy and edit `helloworld.mcp`. If you are using GNU tools, copy and edit the provided *Makefile*. For instructions on accessing the sample files, see "Downloading and working with the PODS sample files" on page 488.

> **Note**
> When developing for Palm OS, be sure to include the line below in your prefix files: `#define __palmos__` Notice that there are two underscores before, and two more after `palmos` in this line. For examples, see Appendix "PODS code samples" on page 487.

Coding without globals on Palm OS

Palm OS system libraries cannot have global or static variables. This is a limitation of Palm OS system libraries. You can define static constants (`const`), because these go into the code resource. However, if you are using CodeWarrior, static constants do not always go into the code resource.

> **Note**
> To determine if you are using global or static variables, check the size of each segment after you build your project. If you are using the Metrowerks IDE, you can check the segment sizes in the Project Inspector window. If you are using GNU tools, use the `objdump` tool. The size of the `.data` and `.bss` segments should be `0` for a Palm OS shared library. If they are not `0`, check your code for global or static data.

Avoiding global data on Palm OS can be difficult. CodeWarrior for Palm OS may store any structure or array declared globally in the global data segment, even if the structure or array is declared `const`. Furthermore, both CodeWarrior and the GCC compiler in the `prc-tools` distribution store any structure in the global data segment if it contains pointers. For example, consider the following structure definition:

```
const struct foo {
  int x;
  const char *s;
  const char *t;
} myfoo = { 4, "mahir", "cagri" };
```


Both CodeWarrior and GCC store this structure in the global data segment, which means that you cannot use this definition in a POD on Palm OS. If you want to use a structure such as this one in a POD on Palm OS, you must generate it from code. The simplest way to do this is to use in-line fixed-size arrays:

```
struct foo {
  int x;
  const char s[8];
  const char t[8];
} myfoo = { 4, "mahir", "cagri" };
```

An alternate way to generate this structure is:

```
typedef struct MyPod {
  PODSPod* pod;
  struct foo myfoo;
} MyPod;

void init(MyPod* pod)
{
  pod->myfoo->x = 4;
  pod->myfoo->s = "mahir";
  pod->myfoo->t = "cagri";
}
```

Other ways of avoiding globals

As illustrated in "Implementing a PODS interface in C" on page 19, your implementation of the `PODSPod` interface can contain private data not defined in the `PODSPod` structure; this is an ideal place to store data that you might otherwise store using global variables in C.

Each function that implements a `PODSPod()` method takes a pointer to a `PODSPod` object as its first argument. In each such function, you can cast this pointer to a pointer to your own structure. That structure can extend the `PODSPod` structure and thus gain access to your globals.

If you are exporting objects to JavaScript and your POD runs on Palm OS, one approach is for each of your custom objects to contain a pointer to your `PODSPod` object. This allows the object's method implementations to gain access to global data that you have stored in the `PODSPod` object.

Building PODS for Microsoft OS devices

On the Microsoft operating systems, you implement a POD as a Windows `.DLL` or dynamic link library. After compiling, you place the `.DLL` in the *Pods* directory under the M-Business Client installation directory on your handheld PC.

♦ To create a POD for a Microsoft OS device

1. Use the Visual C++ wizard to create a new *.dll* project.
2. Include the file, ***podsstartupwin.c***, in the project.
3. Add your own files and methods to the project in C and include *pods.h*, or leverage the ***helloworld.vcp*** project file.

For instructions on locating the sample files, see "Downloading and working with the PODS sample files" on page 488.

4. Place the compiled ***.dll*** in the ***Pods*** directory under the M-Business Client installation directory.

   Uusually this is *\Program Files\AvantGo\Pods*.

> **Note**
> If you want your Microsoft OS code to port to Palm OS without major edits, do not use global or static variables.

Implementing the PODSPodNew( ) function

Every POD must have a `PODSPodNew( )` function that M-Business Client calls to initialize the POD. This is the only external function to the PODS object structure. If `PODSPod` were a C++ class, `PODSPodNew( )` would be comparable to the constructor for your derived class.

There is no default implementation of this function; you must write your own. `PODSPodNew( )` has the following prototype:

```
PODSPod* PODSPodNew(PODSAvantGo* podsavantgo);
```

`PODSPodNew( )` receives a pointer to a `PODSAvantGo` object. The `PODSAvantGo` interface is defined in the *podsavantgo.h* header file. `PODSAvantGo` is the object that manages PODs; its methods provide access to the objects associated with a POD. There is a single `PODSAvantGo` object. See "PODSAvantGo object" on page 83 for details.

`PODSPodNew( )` must return a pointer to an object that implements the `PODSPod` interface. When you write a POD, you can implement a `PODSPod` object using code similar to the code presented in "Implementing a PODS interface in C" on page 19.

Note
In writing the code that implements your POD, be sure that you allocate any `vtable`s with `calloc`. Alternatively, you may allocate `vtable`s with `malloc` and then `memset` them to zero. Failure to do so may leave unassigned `vtable` entries undefined. This can cause unpredictable behavior.

Below is a complete definition of a `PODSPodNew( )` function:

```
PODSPod *PODSPodNew(PODSAvantGo *avantgo)
{
   ObjectSrcPod *self = (ObjectSrcPod *)malloc(sizeof(ObjectSrcPod));
   PODSObjectMgr *objMgr = PODSgetObjectMgr(avantgo);
   PODSObjectSrc *objSrc = (PODSObjectSrc *)ObjectSrcNew(objMgr);

   self->podsPod.avantgo = avantgo;
   self->podsPod.vtable = (PODSPodVTable *)calloc(1, sizeof(PODSPodVTable));
   self->podsPod.vtable->m_getVersion = getVersion;
   self->podsPod.vtable->m_getPodDescription = ObjectSrcPodGetPodDescription;
   self->podsPod.vtable->m_getPodVersion = ObjectSrcPodGetPodVersion;
   self->podsPod.vtable->m_destroy = ObjectSrcPodDestroy;

   PODSregisterObjectSrc(objMgr, objSrc);

   return (PODSPod *)self;
}
```

For more examples of `PODSPodNew( )` implementations, see the sample code listings in Appendix "PODS code samples" on page 487.

Implementing the PODSObject object

All objects in the client extension (PODS) API inherit from the `PODSObject` object. You must implement each of your PODS objects with the three basic `PODSObject` methods:

- "getVersion( )" on page 73 gets the version of M-Business Client whose interface definitions were used to build this object.
- "getMethod( )" on page 70 returns a pointer to an object's method for the specified name.
- "destroy( )" on page 69 frees any internal storage associated with a PODS object.

See "PODSObject object" on page 68.

There is no default implementation of the `PODSObject` methods. You must explicitly implement the `PODSObject` methods in your object's vtable. For most normal objects, the code to implement your `PODSObject` methods would look something like this:

```
vtable->m_getVersion = myGetVersion;
vtable->m_destroy = myDestroy;
vtable->m_getMethod = myGetMethod;
```

For more examples of `PODSObject` object implementations, see the sample code listings in Appendix "PODS code samples" on page 487.

Writing the code for your PODS' main application functionality

All the topics covered so far in this chapter involve setting up the infrastructure for your POD; the goal of this section is to provide some guidance in putting together the code for the heart of a POD, the main functionality for which the POD is being created to support.

Note
In writing the code to implement your POD, be sure that you allocate any `vtable` s with `calloc`. Alternatively, you may allocate `vtable` s with `malloc` and then `memset` them to zero. Failure to do so may leave unassigned `vtable` entries undefined. This can cause unpredictable behavior.

Vending documents: displaying HTML pages

To have your POD display HTML pages, your POD must register a `PODSDocumentSrc` object that implements `documentForUrl()` and/or `documentForSubmission()`. For sample code illustrating the use of these methods, see "DocumentSrc sample: vending documents" on page 491.

These methods are inherited by the `PODSDocumentMgr` object. When your POD executes `documentForUrl()` or `documentForSubmission()` from `PODSDocumentMgr`, these methods examine every registered document source until a match is found: `documentForUrl()` returns a document for that URL; `documentForSubmission()` returns a document for that submission object.

If either `documentForUrl()` or `documentForSubmission()` fail to find a match, they return a `NULL` `PODSDocument` object, along with a `handled` argument set to `PODS_FALSE`. For details on these `PODSDocumentMgr` methods, see:

- "documentForUrl( )" on page 183
- "documentForSubmission( )" on page 182

Note
The `documentForUrl()` and `documentForSubmission()` methods can also set `handled` to `PODS_TRUE` and still return NULL. This is useful if handling the URL or submission does not involve displaying a response. For example, the URL or submission could simply change the state of M-Business Client.

More information on displaying HTML pages follows below. For an example of a POD that submits a form, see the sample code for "Pod sample: submitting forms" on page 489.

Your POD can create a document using `PODSDocumentMgr` object's "createDocument( )" on page 187 and then use the attributes from "PODSObject object" on page 68 to add to it.

The following example uses `createDocument( )` to create a document:


```
static PODSDocument *DocumentSrcDocumentForUrl(
          PODSDocumentSrc *podsDocSrc,
          PODSString url, PODSBoolean *handled)
{
   DocumentSrc *self = (DocumentSrc *)podsDocSrc;

   if (0 == stricmp(url, SAMPLE_URL)) {
      PODSDocument *doc =
           PODScreateDocument(self->documentMgr,
           SAMPLE_URL, PODS_HTML_TYPE);
      ADOMHTMLDocument *dom = PODSgetDom(doc);
      ADOMElement *body;
      ADOMText *node;

      ADOMsetTitle(dom, (ADOMString)"Hello World!");

      body = ADOMcreateElement(dom, (ADOMString)"body");
      ADOMappendChild(dom, (ADOMNode *)body);

      node = ADOMcreateTextNode(dom, (ADOMString)"Hello World!");
      ADOMappendChild(body, (ADOMNode *)node);

      if (handled)
         *handled = PODS_TRUE;

      return doc;
   }

   return NULL;
}
```

Vending objects: exporting your POD to JavaScript

Much of the standard functionality of PODS is automatically available to M-Business JavaScript engine. In order to make your custom POD accessible from M-Business JavaScript engine, there are several things you must do. These tasks are outlined in a separate chapter, "Writing C Code for PODS" on page 13. For an example of a POD that vends an object to JavaScript, see "ObjectSrc sample: vending objects to JavaScript" on page 493.

Naming your PODS objects

A PODS registers a `PODSObjectSrc` with the `PODSObjectMgr`. When an object needs to be looked up, the `PODSObjectMgr` calls `objectForName()` on each `PODSObjectSrc` until an object with a matching name is found. If several objects should happen to have the same name, only the first one found is returned. As a PODS author, you choose the names for your POD's objects. You should follow naming conventions that minimize the likelihood of a name conflict.

iAnywhere Solutions recommends that PODS developers use names of the form `"companyName.privateName"` to name PODS objects. The `"companyName"` portion should be the top-level DNS domain name of the organization to which an author belongs. For example, the company foo.com could use `"foo"` as the `companyName` for naming PODS objects.

Creating multiple PODS for a single domain

If you have multiple PODS for a single domain, you must make sure that each PODS strictly controls the URLs that it handles. There should be no overlap among the URLs that two or more PODS handle. The order in which the PODS receive requests for URLs is unpredictable, so if more than one PODS can handle the same URL there is no way of knowing which PODS actually will handle that URL.

Shortcut for testing PODS

You can always test your POD code by using the procedure outlined in "Installing PODS on users' devices" on page 33, and then synchronizing the test device when you update your POD's compiled binary. If your target platform supports it, there is a faster way.

If your target platform allows you to do any of the following, you can use the testing shortcut detailed below.

- Run M-Business Client in the platform's software emulator
- Browse the Internet directly while the device is in a cradle attached to an Internet-connected PC
- Browse the Internet directly through a wireless connection

In summary, the testing shortcut involves placing your POD's compiled binary on a web server that is accessible to the physical device or software emulator on which you want to test. You can then run the POD by entering its `pods://...` URL in the M-Business Client Open Page dialog on the target device — no synchronization is required.

♦ To test PODS

1. Compile your POD's code.
2. Copy the compiled binary to an accessible web server.

   The web server location must be accessible to the physical target device or software emulator on which you want to test your code.
3. In the M-Business Client **Open Page** dialog on the target physical device or software emulator, enter the URL for the POD.

   The URL is structured as follows:

   `pods://<server_name>/<path>/<binary_name>`

Note
A slight variation on this test approach is to create an HTML page with a link to the POD's compiled binary. The binary can be located on the same server as the HTML page, or it can be on a different server. Use of an intermediate HTML page might allow you to enter a shorter URL into the Open Page dialog, or test several PODS binaries from that page by entering only one URL into the Open Page dialog.

Installing PODS on users' devices

Note
This section presents a summary of the main steps involved in installing PODS on users' devices. For more detailed coverage of all the issues involved, see the *Application Developer Guide for M-Business Anywhere*.

♦ To install a POD on your users' devices

1. Define a channel that includes your POD in its path:
2. Log in to the M-Business Server Administrator Console as **admin**.

   You need to be logged in as Admin in order to enable hiding the channel and synchronizing binaries.
3. Navigate to the **New Channel** page.
4. In the **Location** field, specify a URL to the POD.
5. Set up the web server serving the POD to support MIME type **`application/octet-stream`**.
6. Enable the channel to synchronize binaries.

   Check the **Allow Binary Distribution** channel property. This property enables the channel to synchronize binaries such as a POD.
7. Hide the channel so that it does not appear as a link on the device.

   Check the **Hide from Users** channel property. If this property is not checked, the channel appears in M-Business Client.

CHAPTER 3

Exporting a PODS object to M-Business JavaScript engine

Contents

Introduction to exporting a POD to M-Business JavaScript engine

M-Business JavaScript engine code on HTML pages in M-Business Client can call native code in M-Business client extension API (PODS). Any POD can export objects to M-Business JavaScript engine, and M-Business JavaScript engine code can call `avantgo.createObject()` to get an object exported by a POD. When JavaScript code invokes a method on a PODS object or reads or writes one of the PODS object's properties, M-Business Anywhere client calls native code in the POD that implemented the object.

The `avantgo.createObject()` method is part of the PODS system code that is exported to JavaScript and it is available automatically to JavaScript authors. However, in order for `avantgo.createObject()` to vend objects from your POD, you must do the following:

- **Implement a PODSObjectSrc instance** and register it with the `PODSObjectMgr`. The `objectForName()` method of the `PODSObjectSrc` is used by `avantgo.createObject()`. You do this to make `avantgo.createObject()` vend your object. See "Implementing a PODSObjectSrc instance" on page 37.
- **Implement getMethod()** on the object returned by `objectForName()`. You do this to make calling methods on your vended object work from JavaScript. See "Implementing getMethod( )" on page 39.
- **Register objects and strings** that you do not want to have your object or object source manage directly, so that the memory they use can be freed when no longer needed. See "Registering strings and objects to be freed" on page 40.
- Optionally, **implement a PODSArray instance**, if you want to pass array data between PODS methods and M-Business JavaScript engine. See "Passing array data between PODS methods and M-Business JavaScript engine" on page 41.

> **Note**
> These steps are necessary when you want to vend objects, not when you want to get objects vended by some other POD.

For instructions on accessing your POD from JavaScript, once you have completed the tasks outlined in this chapter, refer to the *M-Business Channel Developer Guide* section titled "Using PODS functions from JavaScript engine" [*M-Business Anywhere Application Developer Guide*]. For an example of a POD that vends an object to JavaScript, see "ObjectSrc sample: vending objects to JavaScript" on page 493.

Implementing a PODSObjectSrc instance

Implement a `PODSObjectSrc` instance:

```
// Create our object source and fill in vtable
// entries for objectForName and destroy
ObjectSrc *ObjectSrcNew(PODSObjectMgr *objectMgr)
{
   ObjectSrc *self = (ObjectSrc *)calloc(1, sizeof(ObjectSrc));

   self->vtable = (PODSObjectSrcVTable *)calloc(1,
sizeof(PODSObjectSrcVTable));

   PODS_SET_METHOD(self->vtable, objectForName, ObjectSrcObjectForName);
   PODS_SET_METHOD(self->vtable, destroy, ObjectSrcDestroy);

   self->objectMgr = objectMgr;

   return self;
}
```

And register it with the `PODSObjectMgr` :

```
PODSregisterObjectSrc(objMgr, objSrc);
```

When JavaScript code calls the JavaScript `avantgo.createObject()` method, M-Business Client calls `PODSObjectMgr`'s `objectForName()` method, which begins calling the `objectForName()` method of every `PODSObjectSrc` instance registered with `PODSObjectMgr`, passing the string that JavaScript passed to `avantgo.createObject()`. `PODSObjectMgr` stops calling registered `objectForName()` methods as soon as one successfully returns an object.

Each object source should handle the `objectForName()` method either by returning a PODS object to JavaScript or by returning `NULL` if the object source cannot return an object with the given name. An object source may handle the `objectForName()` method either by creating a completely new PODS object or by returning an existing object.

An object source may return the same object in response to many different calls to the `objectForName()` method, or it may create a new object for each call. In either case, it must arrange for the object to be freed correctly:

```
static PODSObject
                      *ObjectSrcObjectForName(PODSObjectSrc
                      *podsObjSrc, PODSString name)
{
   ObjectSrc *self = (ObjectSrc *)podsObjSrc;

   if (0 == strcmp(name, SAMPLE_NAME)) {
      if (!self->sampleObject)
         self->sampleObject = SampleObjectNew();

      return (PODSObject *)self->sampleObject;
   }

   return NULL;
}
```


The `objectForName()` method returns a `PODSObject`, so any object returned by the `objectForName()` method must, at the least, implement the methods defined by the `PODSObject` interface. Note that JavaScript code cannot invoke these methods directly. An object returned by the `objectForName()` method may implement additional interfaces if you need it to do so.

The code for the sample object below is from the *sampleobjectsrc.c* file from the `ObjectSrc` example. See "ObjectSrc sample: vending objects to JavaScript" on page 493.

```
SampleObject *SampleObjectNew()
{
   SampleObject *self = (SampleObject *)calloc(1, sizeof(SampleObject));

   self->vtable = (SampleObjectVTable *)calloc(1,
sizeof(SampleObjectVTable));

   PODS_SET_METHOD(self->vtable, add, SampleObjectAdd);
   PODS_SET_METHOD((PODSObjectVTable *)self->vtable, destroy,
SampleObjectDestroy);
   PODS_SET_METHOD((PODSObjectVTable *)self->vtable, getMethod,
SampleObjectGetMethod);

   return self;
}
```

For guidelines on naming PODS objects, see "Naming your PODS objects" on page 30.

Implementing getMethod( )

The `getMethod()` method returns a pointer to a function that implements a PODS method. To implement `getMethod()`, simply compare the requested method's name with each method name the object supports. For example:

```
static PODSMethod
                        SampleObjectGetMethod(PODSObject
                        *podsObj, PODSString methodName,
                        PODSString *methodSignature)
{
   if (0 == strcmp(methodName, "add")) {
       *methodSignature = "ii_i";
       return (PODSMethod)SampleObjectAdd;
   }
   if (0 == strcmp(methodName, "destroy")) {
       *methodSignature = "";
       return (PODSMethod)SampleObjectDestroy;
   }

   return NULL;
}
```

For information on the values that `getMethod()` returns, see "Type strings returned by getMethod( )" on page 71.

Registering strings and objects to be freed

Registering strings

When a PODS method returns a string, the caller should call the `PODSMemoryMgr`'s `stringFree()` method to free the returned string.

When a PODS method returns a string, the implementer of the method may wish to register the string before returning it. Registering a string allows the developer to control what happens when the `PODSMemoryMgr`'s `stringFree()` method is used to free the string. You may use `stringFree()` on an unregistered string, however, doing so will have no effect.

Most PODS authors will probably choose to use `PODSMemoryMgr`'s `stringDupAndRegister()` method to duplicate and register strings. `Calling stringFree()` on a string made with `stringDupAndRegister()` will cause the string to be freed.

When you want finer control over the allocation of returned strings, you may use `stringRegister()`. Along with the string, `stringRegister()` takes a pointer to `PODSFreeFunc` and a void pointer as parameters. Later, when `stringFree()` is called, the `PODSFreeFunc` supplied for the string will be called and will be passed the string as well as the supplied void pointer. This void pointer may be used to store any necessary state. We expect `stringDupAndRegister()` to be sufficient for most PODS authors. But more complicated string management schemes, such as interning or reference counting, could be implemented using `stringRegister()`.

Reference counting: registering and freeing objects

`PODSObject` lifetime is managed by a reference counting mechanism. This mechanism is similar to the reference counting mechanism in Microsoft's COM. The rules are:

- The `PODSObject` implementer should implement the "addRef( )" on page 68 and "release( )" on page 74 methods and call `addRef()` before returning a `PODSObject`.
- When a `PODSObject` is returned, through either a return value or an out parameter (`PODSObject**`), the caller owns the object. The caller is responsible for calling the `release()` method when done with the object.

> **Note**
> The above also applies to arrays created by "createStdArray( )" on page 86.

Passing array data between PODS methods and M-Business JavaScript engine

If you want to pass array data to or return it from PODS methods called by M-Business JavaScript engine, you must implement an instance of the `PODSArray` interface. M-Business Client provides a generic implementation of `PODSArray` that you can access through the `PODSAvantGo` object's "createStdArray( )" on page 86 method. This approach requires filling in each location as a `PODSVariant`.

If the data that you want to return is already an array in the C code, you can improve speed and memory performance by implementing a `PODSArray` object around your data in place. See "PODSArray object" on page 104.

Below is an example of using `createStandardArray()` to create a four-element array:

```
PODSArray *newArray;
PODSVariant *element;

newArray = PODScreateStdArray(podsPod.avantgo, 4);
element = (PODSVariant *) malloc(sizeof(PODSVariant));

element->vt = PODS_STRING;
element->u.strVal = "Data";
PODSsetElement(newArray, 0, element);
element->u.strVal = "String";
PODSsetElement(newArray, 1, element);
element->vt = PODS_INT32;
element->u.i32Val = -9;
PODSsetElement(newArray, 2, element);
element->vt = PODS_UINT32;
element->u.u32Val = 42;
PODSsetElement(newArray, 3, element);

free(element);
return newArray;
}
```


CHAPTER 4

Downloading a POD

Contents

Sequence of events

The following code examples illustrate how to programmatically bring down PODS onto a device. This code is a sample application running on an application server, or web server, behind M-Business Server. The examples include code checks to verify that the correct Palm OS for the device is downloaded.

♦ **To download a POD to a user's device**

1. "Determine the device type and set the MS_DEVICE_TYPE variable" on page 45
2. "Verify that a compatible M-Business browser is being used" on page 47
3. "Use the globally assigned variable to download the correct PODS" on page 48

Determine the device type and set the MS_DEVICE_TYPE variable

The following C# code shows you how to obtain the header information to determine the device type and set the `MS_DEVICE_TYPE` variable.

```
string checkPOS5 = (string) ConfigurationSettings.AppSettings[
"AllowOnlyPalmOS5orGreater" ];
if( null != checkPOS5 ) {
checkPOS5.ToLower();
}
if( null != checkPOS5 && 0 < checkPOS5.Length &&
''t'' == checkPOS5[0] &&
"PALM_OS".Equals((string)Session[ "MS_DEVICE_TYPE" ] )) {

string[] osv =
Request.Headers.GetValues( "X-AvantGo-DeviceOSVersion" );

if( null != osv && 0 < osv.Length ) {
AGLog.Write( AGLog.INFO,
"start.aspx.cs: AllowOnlyPalmOS5orGreater " +
(string)Session[ "MS_DEVICE_TYPE" ] + " " +
AvantGo.Utils.Base64DecodeString( osv[0] ) );
} else {
AGLog.Write( AGLog.INFO,
"start.aspx.cs: AllowOnlyPalmOS5orGreater " +
(string)Session[ "MS_DEVICE_TYPE" ] +
"\nMissing X-AvantGo-DeviceOSVersion header." );
}

bool invalid = true;
if( null != osv && 0 < osv.Length ) {
//String POSver = // a test for 5.0.0
// AvantGo.Utils.Base64DecodeString( "NS4wLjA=" );
String POSver = AvantGo.Utils.Base64DecodeString( osv[0] );
AGLog.Write( AGLog.INFO,
"start.aspx.cs: PalmOS version found: " + POSver);
string [] vers = POSver.Split( ''.'' );
short sver = 0;
if( 1 <= vers.Length ) {
try {
sver = System.Convert.ToInt16( vers[0] );
} catch( Exception ) {
}
}
if( 5 <= sver ) {
invalid = false;
}
}
if( invalid ) {
AGLog.Write( AGLog.WARN,
"start.aspx.cs: Invalid PalmOS found!" );
AGSyncLog.UpdateSyncStatus(syncId,"2000",
"Invalid Palm OS Version",
transactionsQueued,
transactionsProcessed,
transactionsFailed);
Response.Redirect( Request.ApplicationPath +
"/avantgo/invalidclientpage/invaliddeviceos.html" );
}
}
```



```
string[] tmpKey = Request.Headers.GetValues("X-AvantGo-Version");
if (tmpKey != null) {
Session["AvantGo-Version"] = tmpKey[0];
} else {
Session["AvantGo-Version"] = "10.10.1000";
}
```

Verify that a compatible M-Business browser is being used

The following C# code shows you how to verify a compatible M-Business browser is being used.

```
AGLog.Write(AGLog.INFO, "start.aspx.cs: Checking version... :" +
Session["AvantGo-Version"]);

// Require client version 5 or better
int RequiredMajorVersion = 5;
string version = (string)Session["AvantGo-Version"];
if(version != null) {
string [] vers = version.Split(''.'');
if(vers.Length != 3 ||
System.Convert.ToInt16(vers[0]) < RequiredMajorVersion) {
AGLog.Write(AGLog.WARN,
"start.aspx.cs: Invalid client version: " + version
+ "Device Type: " +
(string)Session[ "MS_DEVICE_TYPE" ]);
AGSyncLog.UpdateSyncStatus(syncId,"3000",
"Invalid Client Version",
transactionsQueued,
transactionsProcessed,
transactionsFailed);
Response.Redirect( Request.ApplicationPath +
"/avantgo/invalidclientpage/invalidclientpage.aspx");
return;
}
}

AGLog.Write(AGLog.INFO, "start.aspx.cs: Done checking version...");
}
```

Use the globally assigned variable to download the correct PODS

The following JavaScript shows you how to download the correct PODS for the device.

```
<!– START avantgo_extensions/listpage/ui_footer.inc –>
<br/><br/>

<!– UTIL child links –>
<A HREF="<%=MS_ROOT_URL%>/avantgo/util/util.aspx"></A>
<A HREF="<%=MS_ROOT_URL%>/avantgo/calmonthpage/calmonthpage.aspx"></A>

<!– APPLICATION RESOURCE LINKS –>
<% if(MS_DEVICE_TYPE == "PALM_OS") { %>
<A HREF="<%=MS_ROOT_URL%>/pods/PALM_OS/dbpod_quad.prc"></A>
<A HREF="<%=MS_ROOT_URL%>/pods/PALM_OS/mimelist.prc"></A>
<A HREF="<%=MS_ROOT_URL%>/pods/PALM_OS/xdbset.prc"></A>
<A HREF="<%=MS_ROOT_URL%>/pods/PALM_OS/pimset.prc"></A>
<A HREF="<%=MS_ROOT_URL%>/pods/PALM_OS/mimedtpicker.prc"></A>
<% } else if(MS_DEVICE_TYPE == "WINCE_OS") { %>
<A HREF="<%=MS_ROOT_URL%>/pods/AG_DEVICEOS/AG_DEVICEPROCESSOR/
dbpod_quad.dll"></A>
<A HREF="<%=MS_ROOT_URL%>/pods/AG_DEVICEOS/AG_DEVICEPROCESSOR/
mimelist.dll"></A>
<A HREF="<%=MS_ROOT_URL%>/pods/AG_DEVICEOS/AG_DEVICEPROCESSOR/xdbset.dll"></
A>
<A HREF="<%=MS_ROOT_URL%>/pods/AG_DEVICEOS/AG_DEVICEPROCESSOR/pimset.dll"></
A>
<A HREF="<%=MS_ROOT_URL%>/pods/AG_DEVICEOS/AG_DEVICEPROCESSOR/
mimedtpicker.dll"></A>
<% } %>
```

Part II. Reference

CHAPTER 5

PODS API mechanics

Contents

Roadmap to PODS interfaces

This section summarizes the functionality provided by each M-Business client extension API (PODS) interface. From these summaries, you should be able to determine which interface you need to use to implement which types of application tasks in PODS.

> **Note**
> To quickly determine which PODS objects are exported to M-Business JavaScript engine, refer to "PODS interface inheritance" on page 57.

Object-related and top-level objects:

- **PODSObject** is the basic interface of all PODS objects. The `PODSObject` object is used occasionally in function prototypes to refer to an object with a generic type. See "PODSObject object" on page 68.
- **A PODSObjectSrc** object implements objectForName(), which returns a `PODSObject` for a specified name. See "PODSObjectSrc object" on page 76.
- **The PODSObjectMgr object** manages the `PODSObjectSrc` objects associated with a POD. It checks to see if a `PODSObject` implements a specific interface. See "PODSObjectMgr object" on page 78.
- **PODSAvantGo** object represents the M-Business application. You can get several important PODS objects (`PODSMemoryMgr`, `PODSSubmissionMgr`, `PODSObjectMgr`, etc.) from the `PODSAvantGo` object. See "PODSAvantGo object" on page 83.
- **A PODSPod** object represents a loaded POD. The interface has methods to get the POD's description and version. See "PODSPod object" on page 94.
- **The PODSMemoryMgr** provides memory management tools that are needed by PODs. See "PODSMemoryMgr object" on page 96.
- **A PODSArray** object is an array that is passed between PODS and JavaScript. See "PODSArray object" on page 104.

Document-related objects:

- **A PODSDocument** represents one document in the system, which can be an HTML page, an image, or a script. `PODSDocument` allows you to access all the document attributes. See "PODSObject object" on page 68.
- **A PODSDocumentSrc** vends documents in response to URLs. PODSDocumentSrcs are registered with the `PODSDocumentMgr`. See "PODSDocumentSrc object" on page 180.
- **A PODSDocumentEnumerator** is used to enumerate the documents available from the `PODSDocumentSrc` that vended it. See "PODSDocumentEnumerator object" on page 185.
- **The PODSDocumentMgr** method registers, unregisters, and creates `PODSDocumentSrc` objects. See "PODSDocumentMgr object" on page 187.

DOM-related objects:

- **The ADOMObject** interface includes attributes and methods that implement most of the Worldwide Web Consortium (W3C) document object model (DOM) level 1 specification. See "ADOMDOMImplementation object" on page 110.

Submission-related objects:

- **A PODSSubmissionElement** represents a name-value pair in a `PODSSubmission` object. See "PODSObject object" on page 68.
- **A PODSSubmission** represents a form submission request. It also allows creation of its own submission elements. See "PODSSubmission object" on page 199.
- **The PODSSubmissionMgr** manages the form submissions. You usually use this object only when creating or deleting a `PODSSubmission` object programmatically. See "PODSSubmissionMgr object" on page 218.

Browser-related objects:

- **A PODSButton** represents a button in the M-Business Client toolbar. `PODSButton` is the full interface that is available to PODS to manipulate the M-Business Client toolbar. See "PODSButton object" on page 228.
- **The PODSToolbar** represents the M-Business Anywhere client toolbar. See "PODSToolbar object" on page 232.
- **A PODSWindow** object represents the M-Business Anywhere client browser window. Equivalent to JavaScript `Window` object. See "PODSObject object" on page 68.
- **A PODSHistory** represents the user's browser history and navigates to specified points within that browse history. Equivalent to JavaScript `History` object. See "PODSHistory object" on page 253.
- **The PODSLocation** object represents the location of the page currently being displayed in the M-Business Anywhere client browser. Equivalent to JavaScript `Location` object. See "PODSLocation object" on page 258.

Miscellaneous objects:

- **The PODSScreen** object accesses information about the screen on which the M-Business Anywhere client is running. Equivalent to JavaScript `Screen` object. See "PODSScreen object" on page 288.
- **The PODSPrefs** object represents the set user preferences on M-Business Anywhere client. See "PODSPrefs object" on page 294.
- **The PODSNavigator** object returns information about the M-Business Anywhere client browser application. Equivalent to JavaScript `Navigator` object. See "PODSNavigator object" on page 304.
- **The PODSSymbolScanner** object provides the API for working with a Symbol Technologies bar-code scanner device. See "PODSSymbolScanner object" on page 308.

Event and exception objects:

- **A PODSEventHandler** object handles OS events. See "PODSObject object" on page 68.
- **The PODSEventMgr** manages events. You register and unregister event handlers with the `PODSEventMgr`. See "PODSEventMgr object" on page 322.
- **A PODSException** represents a thrown exception. See "PODSException object" on page 325.
- **The PODSExceptionMgr** provides JavaScript-compatible exception handling for PODS, including the ability to throw an exception to be caught in M-Business JavaScript engine. See "PODSExceptionMgr object" on page 327.

Interface inheritance

A PODS interface can extend another PODS interface, meaning that the interface includes all of the methods of its parent interface(s). For example, you could define an interface `PODSCounterDeluxe` to extend the hypothetical `PODSCounter` interface described above:

```
typedef struct PODSCounterDeluxe PODSCounterDeluxe;
typedef struct PODSCounterDeluxeVTable {
PODSCounterVTable counter;
void (*m_decrement)(PODSCounterDeluxe* c);
} PODSCounterDeluxeVTable;

struct PODSCounterDeluxe {
PODSCounterDeluxeVTable* vtable;
};

#define PODSdeluxeDecrement(c) \
          ((c)->vtable->m_decrement)(c))
```

This interface adds a `decrement` method to the methods defined by the `PODSCounter` interface.

If you have a pointer to an object that implements `I`, you can safely cast to a pointer to an object implementing an interface that `I` extends. For example:

```
PODSCounterDeluxe* d;
...
PODSCounter* c = (PODSCounter* ) d;
```

This cast works because the `PODSCounterDeluxe` and `PODSCounter` structure both contain just a single element: a `vtable` pointer. You could cast `PODSCounterDeluxeVTable*` safely to `PODSCounterVTable*` because `PODSCounterDeluxeVTable` extends `PODSCounterVTable`. A cast from the interface pointer type `PODSCounterDeluxe*` to `PODSCounter*` also works.

Note
Casting in the opposite direction would not be safe. Casting up, to the parent interface, is safe. Casting downward, to the child interface is not.

If interface `I` extends interface `J`, then `J` is a super-interface of `I`. You must cast when calling a method defined in a super-interface, in order to avoid a fatal error or warning (depending on the compiler) when you compile:

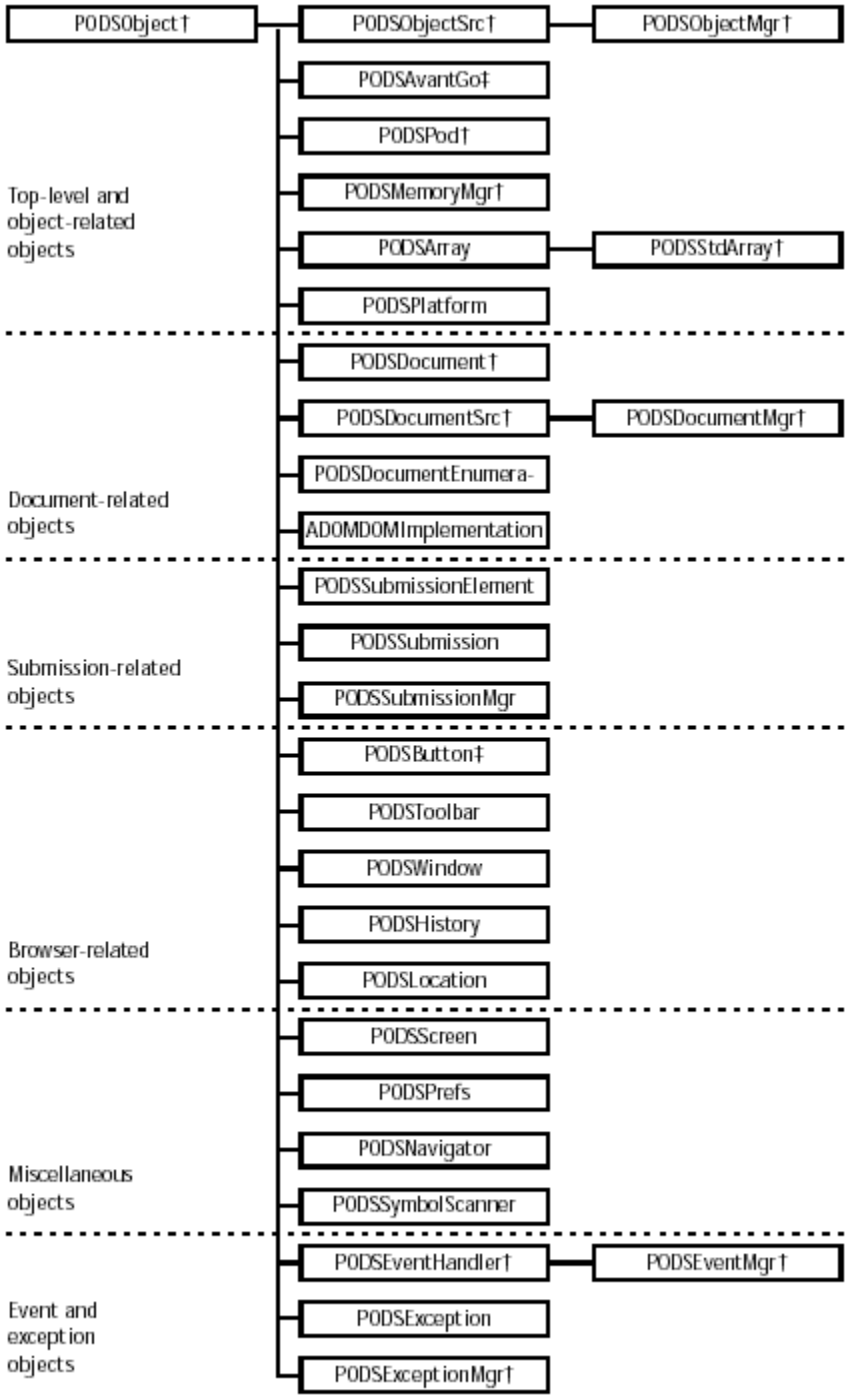
```
PODSCounterDeluxe* d;
...
PODSsetCount((PODSCounter* ) d, 0);
PODSincrement((PODSCounter* ) d);
PODSdeluxeDecrement(d);
```


Note
The PODS object model supports single interface inheritance only: an interface can extend only one other interface directly. This is a simpler model than that provided by the object systems of C++ or Java, both of which allow multiple interface inheritance. The C interfaces provided by the PODS interface header files generally do not use inheritance beyond inheriting from the base interface `PODSObject` described in the following section. Several manager interfaces (`PODSObjectMgr`, `PODSDocumentMgr`, `PODSEventMgr`) and the `PODSButton` interface inherit from `PODSObject` indirectly, via an intermediate interface.

"PODS interface inheritance" on page 57 illustrates PODS interface inheritance graphically.

Note
The `PODSPod` interface is unique in that it also contains data. There is no reason to add methods to a `PODSPod` subclass, but you may want to add more data to a `PODSPod` subclass.

PODS interface inheritance



† indicates object not available in JavaScript engine
‡ indicates object has only select methods/attributes available in JavaScript engine

PODS data types

The PODS data types, defined in the *podstypes.h* file, are described in "PODS data types" on page 58.

Caution
Whenever possible, you should use PODS data types instead of any equivalent data type that may be available in C. By using PODS data types, your code will be insulated from any future changes in the way these data types are defined; it also will make your PODS code more portable.

Table 1. PODS data types

PODS data type*	Description
`PODSArray`	A one-dimensional array of a specified length. The `PODSArray` interface supports array manipulation of the array's individual elements, allowing for passing an array from a JavaScript method into a PODS method, or from a PODS method into a JavaScript method. See "PODSArray object" on page 104.
`PODSBoolean`	A boolean value.
`PODSDate`	A signed 32-bit integer representing a date and time as a count of the number of seconds elapsed since January 1, 1970 UTC. Negative values indicate points in time before January 1, 1970 UTC.
`PODSDouble`	The `PODSDouble` parameter establishes a 64-bit floating-point number.
`PODSErr`	A short value representing a PODS error number.
`PODSInt16`	A 16-bit integer value.
`PODSInt32`	A 32-bit integer value.
`PODSString`	A character string.
`PODSUInt16`	An unsigned 16-bit integer value.
`PODSUInt32`	An unsigned 32-bit integer value.
`PODSUInt8`	An 8-bit unsigned integer.

PODS data type*	Description
PODSVariant	A data type that can accept JavaScript values that may be of any one of several PODS types, without performing any type conversion. PODS data types allowed are: ♦ PODSString ♦ PODSBoolean ♦ PODSDate ♦ PODSDouble ♦ PODSInt32 ♦ PODSObject* ♦ PODSUInt32

* Default values for PODS data types and codes used to represent PODS data types in the type string that PODSObject object's “getMethod( )” on page 70 returns are detailed in “Type strings returned by getMethod( )” on page 71.

Constants to specify a title's character set

Whenever you can specify the character set for a title in PODS, you should use a pre-defined constant from the table below. The following method and attribute, in particular, use these constants to specify the `titleCharset` operand:

- "titleCharset" on page 177
- "createMdbcsSubmission( )" on page 219

Table 2. Title character set constants

Character set constant	Language or language group
AENCODING_ISO8859	Western European
AENCODING_SJIS	Shift-JIS Japanese
AENCODING_ISO8859_2	Eastern European
AENCODING_ISO8859_3	Southern European
AENCODING_ISO8859_4	Northern European
AENCODING_ISO8859_5	Cyrillic
AENCODING_MS1250	Microsoft Central Europe codepage
AENCODING_MS1251	Microsoft Cyrillic codepage
AENCODING_MS1252	Microsoft Latin-1 codepage
AENCODING_KOI8_R	Russian and Ukranian
AENCODING_BIG5	Traditional Chinese (Taiwan)
AENCODING_GB2312	Simplified Chinese (PRC)
AENCODING_UTF_8	Unicode Transmission Format
AENCODING_MS932	Microsoft Shift-JIS codepage
AENCODING_MS936	Microsoft Simplified Chinese
AENCODING_MS949	Microsoft Korean Hangul
AENCODING_MS950	Microsoft Traditional Chinese
AENCODING_EUC_JP	EUC Japanese

Character set constant	Language or language group
AENCODING_EUC_KR	EUC Korean
AENCODING_EUC_CN	EUC Simplified Chinese
AENCODING_EUC_TW	EUC Traditional Chinese
AENCODING_KS_C_5601	Korean National Standard
AENCODING_ISO8859_6	Arabic
AENCODING_ISO8859_7	Greek
AENCODING_ISO8859_8	Hebrew
AENCODING_MS1256	Microsoft Arabic
AENCODING_MS1253	Microsoft Greek
AENCODING_MS1255	Microsoft Hebrew

Deriving C macro method syntax directly from IDL source

Basic approach

Aside from a few notable exceptions, such as the PODS header file, *pods.h*, and *podspod.h*, the source files for PODS are Interface Definition Language (IDL) files. The vast majority of header files in M-Business Anywhere are generated from the IDL files. This API reference documents the details of calling each method in each interface, through the method's associated macro.

As you work with this documentation, the header files, and the IDL files, you will notice that there is a pattern between what you see in the IDL file and what is documented in the C Syntax heading for any particular method. As you become more comfortable with PODS, you may want to try using the IDL files as your primary documentation. In particular, you will find any developer comments in the IDL files, rather than in the generated header files.

Using only the IDL file, you can derive the syntax for calling any given method through its macro by following the steps below. If the IDL definitions are from the W3C DOM spec, see "Differences for ADOMDOMImplementation object methods" on page 64.

♦ **To derive the syntax for calling any given method through its macro**

1. Open the IDL file that contains the method you want to use.

   For an example, we will look at several methods in the *podssubmissionmgr.idl* file.

2. Locate the interface definition section for the interface containing the method you want to use.

   Some IDL files (and corresponding header files) contain multiple interface definitions. In our example, we will look at several methods in the `PODSSubmission` interface. The relevant lines from that file are listed below:

   ```
   ...
   interface PODSSubmission : PODSObject
   {
       attribute PODSString            status;
   ...
       readonly attribute PODSUInt32 submissionElementCount;
   ...
       PODSSubmissionElement
              submissionElementForName(
              PODSString name);
   ...
   ```

3. Locate the line for the method you want to use

   The details of the steps for deriving method syntax from the method's line in the IDL file differ slightly, depending on the way the line appears in the file. The three lines in the example above, from the `PODSSubmission` interface definition in the *podssubmissionmgr.idl* file, illustrate each of these cases.

4. Derive the syntax for the method you want to use as follows:

   If the first term in the IDL file line is `attribute`, then there are two associated methods, a `get` and a `set`. The first method line in our example is of this type:

   ```
   attribute PODSString status
   ```

   - Drop the `attribute` term:

     ```
     PODSString status;
     ```

   - Make the first letter of the method name uppercase and prefix it with `get` (or `set`, as appropriate):

     ```
     PODSString getStatus;
     ```

   - Prefix this method name with `PODS:`

     ```
     PODSString PODSgetStatus;
     ```

   - Add parentheses following the method name:

     ```
     PODSString PODSgetStatus( );
     ```

   - Inside the parentheses, add a pointer to the object named in the object definition statement:

     ```
     PODSString PODSgetStatus(PODSSubmission*);
     ```

   - To the right of the object pointer, add a variable referring to the object itself:

     ```
     PODSString PODSgetStatus(PODSSubmission* sub);
     ```

   If the first term in the IDL file line is `readonly attribute`, then there is only one associated method, a `get`. The second method line in our example is of this type:

   ```
   readonly attribute PODSUInt32 submissionElementCount;
   ```

   - Drop the `readonly attribute` term:

     ```
     PODSUInt32 submissionElementCount;
     ```

   - Make the first letter of the method name uppercase and prefix it with `get`:

     ```
     PODSUInt32 getSubmissionElementCount;
     ```

   - Prefix this method name with `PODS`:

     ```
     PODSUInt32 PODSgetSubmissionElementCount;
     ```

   - Add parentheses following the method name:

     ```
     PODSUInt32 PODSgetSubmissionElementCount( );
     ```

   - Inside the parentheses, add a pointer to the object named in the object definition statement:


```
PODSUInt32 PODSgetSubmissionElementCount(PODSSubmission*);
```

- To the right of the object pointer, add a variable referring to the object itself:

```
PODSUInt32 PODSgetSubmissionElementCount(PODSSubmission* sub);
```

If the first term in the IDL file line is something other than `attribute` or `readonly attribute`, then the associated method is neither a `set` nor a `get`. The third method line in our example is of this type:

```
PODSSubmissionElement submissionElementForName(PODSString name);
```

- Prefix the method name with `PODS` — do not capitalize the first letter of the method name:

```
PODSSubmissionElement PODSsubmissionElementForName(PODSString
name);
```

- Inside the parentheses, add a pointer to the object named in the object definition statement as the very first argument:

```
PODSSubmissionElement
PODSsubmissionElementForName(PODSSubmission* PODSString name);
```

- To the right of the object pointer, add a variable referring to the object itself, followed by a comma:

```
PODSSubmissionElement
PODSsubmissionElementForName(PODSSubmission* sub, PODSString
name);
```

5. If there are additional arguments to the right of the object pointer (not present in the three examples above), there are two additional changes that you may need to make in order to have the correct syntax:

   - If no variable is supplied in the IDL file, as is the case with attributes, you must add one.
   - If the argument is an object, you must make the object name into a pointer by adding an asterisk (*) to its right. This is only true for types specified as `interface` at the top of the file (or in included files). If the type is specified as `typedef`, an asterisk is not required.

Differences for ADOMDOMImplementation object methods

Most of the M-Business "ADOMDOMImplementation object" on page 110 interface implements methods derived from the Worldwide Web Consortium (W3C) Document Object Model (DOM) spec.

For all methods in the `ADOMObject` interface, the instructions in the previous section, "Deriving C macro method syntax directly from IDL source" on page 62, apply with one difference: the prefix you add to the method name is `ADOM` instead of `PODS`.

PODSPodNew( ) function reference

PODSPodNew( ) is the single entry point to the shared library containing a POD. It is comparable to a constructor for a C++ class.

Because PODSPodNew( ) is not part of any PODS object, its reference documentation is located here, in front of the reference documentation on PODS objects which begins with "PODSObject object" on page 68.

There is no default implementation of PODSPodNew( ) ; you must write your own code to implement it. For guidelines on creating your PODSPodNew( ), see "Implementing the PODSPodNew( ) function" on page 27.

PODSPodNew( )

Creates and returns a new PODSPod object. Allocates memory for the POD and tells M-Business Client about this POD. Your implementation may also perform any other initialization functions that your POD requires.

Interface

Not part of any interface

IDL definition

Not applicable

JavaScript synopsis

Not applicable

C synopsis

PODSPod* **PODSPodNew(**PODSAvantGo* *podsavantgo***)**

Parameters

- **podsavantgo** The PODSAvantGo object.

Returns

None

Remarks

This is a required function that must allocate enough memory for the PODSPod structure. PODSPodNew( ) is where a POD would normally register any document sources, object sources, and event handlers. You may perform any other necessary initialization tasks in this function. For example, if your server routinely accesses a database, you might set up access to that database in this method. The simplest way of allocating memory is to make this call:

```
PODSPod* pod = (PODSPod *) MemPtrNew(sizeof(PODSPod));
```


Your POD may require more memory than the PODSPod structure provides. For example, you might need to store the handle of an open database that the POD uses. In this case, you can subclass PODSPod by defining a structure that contains both the PODSPod and the extra fields that you need:

```
typedef MyPod
{
    PODSPod base;
    UInt    dbHandle;
};

....
PODSPod* podSPodNew(PODSAvantGo*podsavantgo) {
    MyPod *pod = (MyPod *)MemPtrNew(sizeof(MyPod));
   ....
    return (PODSPod *)pod;
}
```

To set up the POD's function table, see "Implementing the PODSObject object" on page 28.

See also

PODSObject object's "destroy( )" on page 69

CHAPTER 6

PODS object-related and top level objects

Contents

PODSObject object

- **Inherits from:** N/A
- **Accessed by:** inheritance by other objects
- **Available to:** C only (directly); C and JavaScript via inheritance

All `PODSObject` objects implement the PODSObject interface.

Table 1. Summary of PODSObject attributes and methods

Description	Attributes and methods
Querying PODS object	“getInterface( )” on page 69 “getMethod( )” on page 70 “getVersion( )” on page 73 “nextProperty( )” on page 74
Managing PODS objects	“addRef( )” on page 68 “destroy( )” on page 69 “release( )” on page 74

addRef( )

Adds a reference.

Interface

PODSObject

IDL definition

void addRef( );

JavaScript synopsis

Not applicable

C synopsis

void **PODSaddRef(**PODSObject* *podsobj*);

Parameters

- **podsobj** The `PODSObject` object.

Returns

None

Remarks

For reference counting, use `addRef( )` to add a reference to the object, so that `release( )` can free the allocated memory when it is no longer needed.

See also

"addRef( )" on page 68, "release( )" on page 74

destroy( )

Frees any internal storage associated with a PODS object.

Interface

PODSObject

IDL definition

void destroy( );

JavaScript synopsis

Not applicable

C synopsis

void **PODSdestroy(**PODSObject* *podsobj*);

Parameters

- **podsobj** The `PODSObject` object.

Returns

None

Remarks

You should not call the `destroy( )` method on any M-Business browser objects; to do so may cause M-Business Client to stop working or behave unpredictably.

getInterface( )

Returns a pointer to a `PODSInterface` structure which contains information about the interface that this object implements.

Interface

PODSObject

IDL definition

PODSInterface getInterface( );

JavaScript synopsis

Not applicable

C synopsis

PODSInterface* **PODSgetInterface(**PODSObject* *podsobj*);

Parameters

- **podsobj** The `PODSObject` object.

Returns

Pointer to a `PODSInterface` structure.

getMethod( )

Returns a pointer to an object's method for the specified name.

Interface

PODSObject

IDL definition

```
PODSMethod getMethod(
  PODSString name,
  out PODSString type
);
```

JavaScript synopsis

Not applicable

C synopsis

```
PODSMethod PODSgetMethod(
  PODSObject* podsobj,
  PODSString name,
  PODSString* type
);
```

Parameters

- **podsobj** The `PODSObject` object.
- **name** [in] Name of object's method.
- **type** [out] Type string specifying number of arguments, and data types of the arguments and the return value. See "Type strings returned by getMethod( )" on page 71 and topics that follow.

Returns

Pointer to method for specified name.

Remarks

Useful if a PODS author wants to invoke a method directly, especially from JavaScript. If your POD returns an object that you want to be accessible from JavaScript, your object *must* implement getMethod( ).

See also

"How M-Business Client uses getMethod( )" [*M-Business Anywhere Application Developer Guide*]

Type strings returned by getMethod( )

Parameter type strings returned by "getMethod( )" on page 70 consist of:

- One character indicating the type of each method parameter
- An underscore (_)
- A character indicating the type of the return value

If a method returns no value, the underscore may be omitted.

The characters that indicate method parameter types are listed in the table below. The data types represented are detailed in the sections that follow.

Table 2. Characters indicating method parameter types

Character	PODS data type*	Default Value
a	PODSArray	NULL
i	PODSInt32	0
I	PODSInt16	0
s	PODSString (not NULL)	""
S	PODSString (may be NULL)	NULL
b	PODSBoolean	PODS_FALSE
B	PODSBoolean	PODS_TRUE
o	PODSObject	NULL
d	PODSDate	0
f	PODSDouble	0.0
u	PODSUInt32	0
U	PODSUInt16	0

Character	PODS data type*	Default Value
v	PODSVariant	NULL
V	Variable number of arguments	

* For more information on these data types, see "PODS data types" on page 58.

Below are some examples of parameter type strings that could be returned by `getMethod( )`:

- `"ss_i"` indicates that a method takes two `PODSString` parameters and returns a `PODSInt32`.
- `"_o"` indicates that a method takes no parameters and returns a `(PODSObject *)`.
- `"sib"` indicates that a method takes a `PODSString`, a `PODSInt32`, and a `PODSBoolean` as parameters and returns no value (the method call returns the undefined value to JavaScript).

> **Note**
> Remember that a C function that implements a PODS method takes, as its first parameter, a pointer to the object on which method is being invoked. This pointer is not considered a method parameter and does not appear in the type string.

Variable number of arguments in PODS

The PODS variable type string `V` must be the last parameter type in a type string. The `V` type assumes that the last two parameter types of the PODS method are `PODSUInt16` and `PODSVariant`. `PODSUInt16` tells the actual number of arguments; `PODSVariant` is an array that contains all of the remaining arguments.

Optional parameters

A type string may contain a `"/"` to indicate that any remaining parameters are optional. For example, suppose that a method `foo`'s type string is `"ss/ib"`. The method could be implemented in C using a function such as:

```
void foo(PODSObject* self, PODSString s, PODSString t, PODSInt32 i,
PODSBoolean b)
```

Note that the C function has no special knowledge of the fact that its last two arguments are optional when called from JavaScript. If JavaScript calls

```
p.foo("hello", "world")
```

then M-Business automatically passes the values 0 and `PODS_FALSE` as the method's third and fourth arguments. Similarly, JavaScript can call

```
p.foo("hello", "world", 234)
```

and M-Business supplies the default value `PODS_FALSE` for the fourth argument only. If JavaScript calls


```
p.foo("hello")
```

then a JavaScript error results, because the call has not supplied all required arguments.

> **Note**
> The `b` and `B` parameter types differ only in their default values. If a parameter is required, then `b` and `B` are equivalent.

JavaScript type conversions

If possible, JavaScript converts each parameter passed to a PODS method into the type the method expects. When a parameter's type is `s`, the method expects a non-null string, and so JavaScript converts the JavaScript value null to the string `"NULL"`, then converts the undefined value to the string `"UNDEFINED"`. When a parameter's type is `S`, the method expects a string that may be `NULL`. In that case, if JavaScript code passes either of the JavaScript values null or undefined for a parameter, when M-Business Client calls the underlying PODS method it passes a null pointer for the parameter.

getVersion( )

Gets the version of M-Business Client whose interface definitions were used to build this object.

Interface

PODSObject

IDL definition

PODSUInt16 getVersion( );

JavaScript synopsis

Not applicable

C synopsis

PODSUInt16 **PODSgetVersion(**PODSObject* *podsobj*);

Parameters

- **podsobj** The `PODSObject` object.

Returns

A `PODSUInt16` value representing the version of M-Business Anywhere client whose interface definitions were used to build this object.

Remarks

Useful to ensure compatibility when two PODs are communicating with each other. For objects defined by M-Business, the `GetVersion( )` method always returns the current M-Business Client version number, defined by the constant `PODS_VERSION` in *pods.h*.

nextProperty( )

Gets the next property for the `PODSObject`.

Interface

PODSObject

IDL definition

PODSString nextProperty(any *closure*);

JavaScript synopsis

Not applicable

C synopsis

PODSString **PODSnextProperty(**
  PODSObject* *podsobj***,**
  void* *closure*
);

Parameters

- **podsobj** The `PODSObject` object.
- **closure** [in] Pointer to a storage of 8 bytes zero initialized before it is called upon the first time.

Returns

The name of the next property for the `PODSObject`; null when there are no more properties.

release( )

Releases a reference.

Interface

PODSObject

IDL definition

void release( );

JavaScript synopsis

Not applicable

C synopsis

void **PODSrelease(**PODSObject* *podsobj***);**

Parameters

- **podsobj** The `PODSObject` object.

Returns

None

Remarks

For reference counting, use `release( )` to free the allocated memory when it is no longer needed. A reference to the object first must have been added by `addRef( )`.

See also

"addRef( )" on page 68

PODSObjectSrc object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSAvantGo` object's "createObject( )" on page 85, then `PODSObjectMgr` object's "registerObjectSrc( )" on page 80
- **Available to:** C only

`PODSObjectSrc`'s single method, `objectForName( )`, gets a `PODSObject` given its name.

There are two different cases possible where `objectForName( )` may be used:

1. As **caller**: where the PODS author needs to get an object manager and call one of its methods.
2. As **callee**: where the PODS author is implementing a custom POD and needs to have its methods called.

Table 3. Summary of PODSObjectSrc attributes and methods

Description	Attributes and methods
Getting an object for a specified name	"objectForName( )" on page 76

objectForName( )

Gets the object for a specific name.

Interface

PODSObjectSrc

IDL definition

PODSObject objectForName(PODSString *name*);

JavaScript synopsis

Not applicable

C synopsis

```
PODSObject* PODSobjectForName(
  PODSObjectSrc* objsrc,
  PODSString name
);
```

Parameters

- **objsrc** The `PODSObjectSrc` object.
- **name** [in] Name of object to return.

Returns

`PODSObject` for specified name.

Remarks

Useful when two PODs are communicating with each other. Typically only used from its `PODSObjectMgr` subclass, but should implemented from `PODSObjectSrc` interface if you are implementing your own `PODSObjectSrc`.

The `PODSObjectSrc` implementation of `objectForName( )` only checks the `PODSObjectSrc` on which it is invoked, while the `PODSObjectMgr` implementation walks through all the registered PODS objects until an object matching the specified name is found.

See also

`PODSObjectMgr` object's "objectForName( )" on page 80

PODSObjectMgr object

- **Inherits from:** `PODSObjectSrc`
- **Accessed by:** `PODSAvantGo` object's "objectMgr" on page 90
- **Available to:** C only

The `PODSObjectMgr` works with M-Business JavaScript engine's `avantgo.createObject( )` and manages the objects associated with PODs. It registers and unregisters `PODSObjectSrc` objects, and has a method to let you determine whether a `PODSObject` implements a specific interface.

Typically `objectForName( )`, inherited from `PODSObjectSrc`, is called from `PODSObjectMgr`. See "PODSObjectSrc object" on page 76.

Table 4. Summary of PODSObjectMgr attributes and methods

Description	Attributes and methods
Registering/unregistering object sources	"registerObjectSrc( )" on page 80 "unregisterObjectSrc( )" on page 81
Querying object sources	"implements( )" on page 78 "interfaces" on page 79
Getting an object for a specified name	"objectForName( )" on page 80

implements( )

Determines whether a specified `PODSObject` implements a specified `PODSInterface`.

Interface

PODSObjectMgr

IDL definition

```
PODSBoolean implements(
  PODSObject pobj,
  PODSInterface intf
);
```

JavaScript synopsis

Not applicable

C synopsis

```
PODSBoolean PODSimplements(
  PODSObjectMgr* objmgr,
```



```
  PODSObject* pobj,
  PODSInterface* intf
);
```

Parameters

- **objmgr** The `PODSObjectMgr` object.
- **pobj** [in] The `PODSObject` object to query.
- **intf** [in] The `PODSInterface` object to determine if *pobj* implements.

Returns

`PODS_TRUE` if the `PODSObject` implements the specified `PODSInterface`.

`PODS_FALSE` otherwise.

See also

"interfaces" on page 79

interfaces

A list of all interfaces that all PODS objects implement.

Interface

PODSObjectMgr

IDL definition

readonly attribute PODSInterfaces interfaces;

JavaScript synopsis

Not applicable

C synopsis

PODSInterfaces* **PODSgetInterfaces(**PODSObjectMgr* *objmgr***);**

Parameters

- **objmgr** The `PODSObjectMgr` object.

Returns

List of implemented interfaces.

See also

"implements( )" on page 78

objectForName( )

Gets the object for a specific name.

Interface

PODSObjectMgr

IDL definition

PODSObject objectForName(PODSString *name*);

JavaScript synopsis

Not applicable

C synopsis

```
PODSObject* PODSobjectForName(
  PODSObjectMgr* objmgr,
  PODSString name
);
```

Parameters

- **objmgr** The `PODSObjectMgr` object.
- **name** [in] Name of object to return.

Returns

`PODSObject` for specified name.

Remarks

The `PODSObjectMgr` implementation of `objectForName( )` walks through all the registered PODS objects until an object matching the specified name is found, while the `PODSObjectSrc` implementation only checks the specified PODS object.

See also

`PODSObjectSrc` object's "objectForName( )" on page 76

registerObjectSrc( )

Register a `PODSObjectSrc` object.

Interface

PODSObjectMgr

IDL definition

void registerObjectSrc(PODSObjectSrc *objectSrc*);

JavaScript synopsis

Not applicable

C synopsis

```
void PODSregisterObjectSrc(
  PODSObjectMgr* objmgr,
  PODSObjectSrc* objectSrc
);
```

Parameters

- **objmgr** The `PODSObjectMgr` object.
- **objectSrc** The `PODSObjectSrc` object to register.

Returns

None

Remarks

Used if you are implementing your own `PODSObjectSrc` object. The `PODSObjectMgr` takes ownership of the registered object, so there is no need to unregister it when the application exits. You can, however, unregister the object at any point in execution to free up resources or prevent an object from being accessed again.

See also

"unregisterObjectSrc( )" on page 81

unregisterObjectSrc( )

Register a PODSObjectSrc object.

Interface

PODSObjectMgr

IDL definition

void unregisterObjectSrc(PODSObjectSrc *objectSrc*);

JavaScript synopsis

Not applicable

C synopsis

```
void PODSunregisterObjectSrc(
  PODSObjectMgr* objmgr,
  PODSObjectSrc* objectSrc
);
```

Parameters

- **objmgr** The `PODSObjectMgr` object.
- **objectSrc** The `PODSObjectSrc` object to register.

Returns

None

Remarks

Used if you are implementing your own `PODSObjectSrc` object to free up resources or prevent an object from being accessed again. The `PODSObjectMgr` takes ownership of the registered object, so there is no need to unregister it when the application exits.

See also

"registerObjectSrc( )" on page 80

PODSAvantGo object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSpodNew( )` function; see "Implementing the PODSPodNew( ) function" on page 27
- **Available to:** C (fully), JavaScript (selectively)

The `PODSAvantGo` object represents the running M-Business Client. It provides access to the objects associated with a POD, as well as creating arrays that can be passed to JavaScript and string exceptions that are needed by the POD.

The `PODSAvantGo` methods also manage processes that support synchronizing with the associated M-Business Server. Synchronizing is the process in which forms that have been submitted while a user is offline are sent to the server to be processed.

Table 5. Summary of PODSAvantGo attributes and methods

Description	Attributes and methods
Accessing associated objects	"documentMgr" on page 87 "eventMgr" on page 87 "exceptionMgr" on page 88 "memoryMgr" on page 89 "mimeMgr" on page 90 "objectMgr" on page 90 "platform" on page 91 "preferences" on page 91 "submissionMgr" on page 92 "window" on page 93
Creating associated objects	"createObject( )" on page 85 "createStdArray( )" on page 86
Managing connection	"connect( )" on page 84 "disconnect( )" on page 86 "isOnline( )" on page 88

Description	Attributes and methods
Managing synchronization	"beginSync( )" on page 84 "resetChannels( )" on page 92

beginSync( )

Begins a synchronization with the associated M-Business Server.

Interface

PODSAvantGo

IDL definition

void beginSync( );

JavaScript synopsis

avantgo.**beginSync( )**

C synopsis

void **PODSbeginSync(**PODSAvantGo* *avantgo*);

Parameters

- **avantgo** The `PODSAvantGo` object.

Returns

None

See also

"connect( )" on page 84,"resetChannels( )" on page 92, "isOnline( )" on page 88

connect( )

Initiates a network connection.

Interface

PODSAvantGo

IDL definition

void connect( );

JavaScript synopsis

avantgo.**connect( )**

C synopsis

void **PODSconnect(**PODSAvantGo* *avantgo*);

Parameters

- **avantgo** The `PODSAvantGo` object.

Returns

None

Remarks

Use "isOnline( )" on page 88 after using `connect( )` to determine if the connection succeeded.

See also

"beginSync( )" on page 84, "disconnect( )" on page 86, "isOnline( )" on page 88

createObject( )

Returns a `PODSObject` exported by a POD currently loaded in M-Business. This method queries all loaded PODs. It passes the name to each POD and asks if it can provide an object in return. It returns a `PODSobject`, or `NULL` if no POD can provide an object with the specified name.

Interface

PODSAvantGo

IDL definition

PODSObject createObject(PODSString *name*);

JavaScript synopsis

window.**createObject(***name*)

C synopsis

PODSObject* **PODSCreateObject(**
PODSWindow* *window*,
PODSString *name*
);

Parameters

- **window** The `PODSWindow` object.
- **name** [in] The name of the `PODSObject` to be returned.

Returns

`PODSObject` corresponding to the specified name.

NULL

If no POD can provide an object with the specified name.

See also

"avantgo" on page 239

createStdArray( )

Creates a standard array that can be passed to JavaScript.

Interface

PODSAvantGo

IDL definition

nometadata PODSArray createStdArray(PODSUInt32 *size*);

JavaScript synopsis

Not applicable

C synopsis

```
PODSArray* PODScreateStdArray(
  PODSAvantGo* avantgo,
  PODSUInt32 size
);
```

Parameters

- **avantgo** The `PODSAvantGo` object.
- **size** [in] Size of the array to be created.

Returns

Standard array of specified *size*.

Remarks

Methods to manipulate the array before passing it to JavaScript are provided by the `PODSArray` object. After using `createStdArray( )`, you need to call the array setter to initialize the array. See "Indexed properties" [*M-Business Anywhere Application Developer Guide*].

See also

"PODSArray object" on page 104

disconnect( )

Disconnects from the network.

Interface

PODSAvantGo

IDL definition

void disconnect( );

JavaScript synopsis

avantgo.**disconnect( )**

C synopsis

void **PODSdisconnect(**PODSAvantGo* *avantgo*);

Parameters

♦ **avantgo** The `PODSAvantGo` object.

Returns

None

See also

"connect( )" on page 84, "isOnline( )" on page 88

documentMgr

The associated `PODSDocumentMgr` object.

Interface

PODSAvantGo

IDL definition

nometadata readonly attribute PODSDocumentMgr documentMgr;

JavaScript synopsis

Not applicable

C synopsis

PODSDocumentMgr* **PODSgetDocumentMgr(**PODSAvantGo* *avantgo*);

Parameters

♦ **avantgo** The `PODSAvantGo` object.

Returns

The associated `PODSDocumentMgr` object.

eventMgr

The associated `PODSEventMgr` object.

Interface

PODSAvantGo

IDL definition

nometadata readonly attribute PODSEventMgr eventMgr;

JavaScript synopsis

Not applicable

C synopsis

PODSDocumentMgr* **PODSgetEventMgr(**PODSAvantGo* *avantgo*);

Parameters

- **avantgo** The `PODSAvantGo` object.

Returns

The associated `PODSEventMgr` object.

exceptionMgr

The associated `PODSExceptionMgr` object.

Interface

PODSAvantGo

IDL definition

nometadata readonly attribute PODSExceptionMgr exceptionMgr;

JavaScript synopsis

Not applicable

C synopsis

PODSDocumentMgr* **PODSgetExceptionMgr(**PODSAvantGo* *avantgo*);

Parameters

- **avantgo** The `PODSAvantGo` object.

Returns

The associated `PODSExceptionMgr` object.

isOnline( )

Determines whether M-Business Client is online.

Interface

PODSAvantGo

IDL definition

PODSBoolean isOnline( );

JavaScript synopsis

avantgo.**isOnline( )**

C synopsis

PODSBoolean **PODSisOnline(**PODSAvantGo* *avantgo*);

Parameters

- **avantgo** The `PODSAvantGo` object.

Returns

`PODS_TRUE` if M-Business Client is online.

`PODS_FALSE` otherwise.

See also

"connect( )" on page 84, "beginSync( )" on page 84

memoryMgr

The associated `PODSMemoryMgr` object.

Interface

PODSAvantGo

IDL definition

nometadata readonly attribute PODSMemoryMgr memoryMgr;

JavaScript synopsis

Not applicable

C synopsis

PODSDocumentMgr* **PODSgetMemoryMgr(**PODSAvantGo* *avantgo*);

Parameters

- **avantgo** The `PODSAvantGo` object.

Returns

The associated `PODSMemoryMgr` object.

mimeMgr

The associated `AMIMEManager` object.

Interface

PODSAvantGo

IDL definition

nometadata readonly attribute AMIMEManager mimeMgr;

JavaScript synopsis

Not applicable

C synopsis

AMIMEManager* **PODSgetMimeMgr(**PODSAvantGo* *avantgo*);

Parameters

- **avantgo** The `PODSAvantGo` object.

Returns

The associated `AMIMEManager` object.

objectMgr

The associated `PODSObjectMgr` object.

Interface

PODSAvantGo

IDL definition

nometadata readonly attribute PODSObjectMgr objectMgr;

JavaScript synopsis

Not applicable

C synopsis

PODSDocumentMgr* **PODSgetObjectMgr(**PODSAvantGo* *avantgo*);

Parameters

- **avantgo** The `PODSAvantGo` object.

Returns

The associated `PODSObjectMgr` object.

platform

Gets the `PODPlatform` object for the `PODSAvantGo` object.

Interface

PODSAvantGo

IDL definition

nometadata readonly attribute PODSPlatform platform;

JavaScript synopsis

Not applicable

C synopsis

PODSPlatform* **PODSgetPlatform(**PODSAvantGo* *avantgo*);

Parameters

- **avantgo** The `PODSAvantGo` object.

Returns

The `PODSPlatform` object.

See also

"PODSPlatform object" on page 107

preferences

The `PODSPrefs` object for the `PODSAvantGo` object.

Interface

PODSAvantGo

IDL definition

readonly attribute PODSPrefs preferences;

JavaScript synopsis

avantgo.**preferences**

C synopsis

PODSPrefs* **PODSgetPreferences(**PODSAvantGo* *avantgo*);

Parameters

- **avantgo** The `PODSAvantGo` object.

Returns

The PODSPrefs object for the `PODSAvantGo` object.

See also

"submissionMgr" on page 92

resetChannels( )

Resets M-Business Client channels.

Interface

PODSAvantGo

IDL definition

void resetChannels( );

JavaScript synopsis

avantgo.**resetChannels( )**

C synopsis

void **PODSresetChannels(**PODSAvantGo* *avantgo***);**

Parameters

- **avantgo** The `PODSAvantGo` object.

Returns

None

Remarks

Execute `resetChannels( )` before `beginSync( )` to perform a full synchronization. Execute `beginSync( )` without `resetChannels( )` to perform an incremental synchronization.

Example

```
<html>
<form name=foo>
<input type=button
     onClick="javascript:avantgo.resetChannels();"
     value="Click me to eliminate all WebToGo content
     from this server">
/form
</html>
```

submissionMgr

Gets the `PODSSubmissionMgr` object for the `PODSAvantGo` object.

Interface

PODSAvantGo

IDL definition

readonly attribute PODSSubmissionMgr submissionMgr;

JavaScript synopsis

avantgo.**submissionManager**

C synopsis

PODSSubmissionMgr* **PODSgetSubmissionMgr(**PODSAvantGo* *avantgo***);**

Parameters

- **avantgo** The `PODSAvantGo` object.

Returns

The `PODSSubmissionMgr` object for the `PODSAvantGo` object.

window

The associated `PODSWindow` object.

Interface

PODSAvantGo

IDL definition

nometadata readonly attribute PODSWindow window;

JavaScript synopsis

Not applicable

C synopsis

PODSDocumentMgr* **PODSgetWindow(**PODSAvantGo* *avantgo***);**

Parameters

- **avantgo** The `PODSAvantGo` object.

Returns

The associated `PODSWindow` object.

PODSPod object

- **Inherits from:** `PODSObject`
- **Accessed by:** writing your own, using *podspod.h* template
- **Available to:** C only

The `PODSPod` methods get a POD's description or version.

Unlike all the other public components of PODS, `PODSPod` is not a pure interface. `PODSPod` contains data, and subclasses are expected to use implementation inheritance rather than interface inheritance.

The *podspod.h* file is the only interface header file in the PODS directory which is not generated from an interface definition language (IDL) source.

All PODs will subclass `PODSPod`. See the sample PODs for various techniques.

Table 6. Summary of PODSPod attributes and methods

Description	Attributes and methods
Getting POD description or version	"getPodDescription( )" on page 94 "getPodVersion( )" on page 95

getPodDescription( )

Gets the POD description.

Interface

PODSPod

IDL definition

Not applicable

JavaScript synopsis

Not applicable

C synopsis

PODSString **PODSgetPodDescription(**PODSPod* *ppod***);**

Parameters

- **ppod** The `PODSPod` object.

Returns

POD description.

Remarks

Called to get description for display in application About box. Useful for returning a string-based description of your POD.

See also

"getPodVersion( )" on page 95

getPodVersion( )

Gets the POD version.

Interface

PODSPod

IDL definition

Not applicable

JavaScript synopsis

Not applicable

C synopsis

PODSString **PODSgetPodVersion(**PODSPod* *ppod*);

Parameters

- **ppod** The PODSPod object.

Returns

POD version.

Remarks

Called to get version for display in application About box.

See also

"getPodDescription( )" on page 94

PODSMemoryMgr object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSAvantGo` object's "memoryMgr" on page 89
- **Available to:** C only

The `PODSMemoryMgr` methods provide memory management tools that are needed by all PODs.

Table 7. Summary of PODSMemoryMgr attributes and methods

Description	Attributes and methods
Managing memory	"smMalloc( )" on page 100 "smFree( )" on page 99 "smWrite( )" on page 102 "smMemCopy( )" on page 100 "smDup( )" on page 98 "smStrDup( )" on page 101
Registering or freeing strings or objects	"stringDupAndRegister( )" on page 96 "stringFree( )" on page 97 "stringRegister( )" on page 98

stringDupAndRegister( )

Duplicates and registers a string.

Interface

PODSMemoryMgr

IDL definition

```
PODSString stringDupAndRegister(PODSString);
```

JavaScript synopsis

Not applicable

C synopsis

```
PODSString stringDupAndRegister(
  PODSMemoryMgr* memmgr,
  PODSString str
);
```


Parameters

- **memmgr** The `PODSMemoryMgr` object.
- **str** [in] String to duplicate and register.

Returns

Duplicated string.

See also

"smStrDup( )" on page 101

"stringRegister( )" on page 98

stringFree( )

Frees a string.

Interface

PODSMemoryMgr

IDL definition

void stringFree(PODSString *str*);

JavaScript synopsis

Not applicable

C synopsis

void **PODSstringFree(**
  PODSMemoryMgr* *memmgr*,
  PODSString *str*
);

Parameters

- **memmgr** The `PODSMemoryMgr` object.
- **str** [in] String to free.

Returns

None

See also

"stringRegister( )" on page 98

stringRegister( )

Registers a string.

Interface

PODSMemoryMgr

IDL definition

```
void stringRegister(
  PODSString str,
  PODSFreeFunc ff,
  ObjectLifeTime lt
);
```

JavaScript synopsis

Not applicable

C synopsis

```
void PODSstringRegister(
  PODSMemoryMgr* memmgr,
  PODSString str,
  PODSFreeFunc ff,
  ObjectLifeTime lt
);
```

Parameters

- **memmgr** The `PODSMemoryMgr` object.
- **str** [in] String to register.
- **ff** [in] Pointer to PODS free function to use to free the string.
- **lt** [in] Object lifetime of the string, from *podstypes.h*:

```
typedef enum ObjectLifeTime {
  AnyTime,
  EventTime,
  PageTime,
  ApplicationTime
 } ObjectLifeTime;.
```

Returns

None

See also

"stringFree( )" on page 97

smDup( )

Duplicate a range of memory as a constant.

Interface

PODSMemoryMgr

IDL definition

const any smDup(any *ptr*, PODSUInt32 *size*);

JavaScript synopsis

Not applicable

C synopsis

```
const void* PODSsmDup(
  PODSMemoryMgr* memmgr,
  void* ptr,
  PODSUInt32 size
);
```

Parameters

- **memmgr** The `PODSMemoryMgr` object.
- **ptr** [in] Pointer to the start position in memory to duplicate.
- **size** [in] Number of bytes to duplicate.

Returns

Copy of the specified range of memory.

See also

"smMemCopy( )" on page 100, "smStrDup( )" on page 101

smFree( )

Free a range of memory.

Interface

PODSMemoryMgr

IDL definition

void smFree(any *ptr*);

JavaScript synopsis

Not applicable

C synopsis

```
void PODSsmFree(
  PODSMemoryMgr* memmgr,
  void* ptr
);
```


Parameters

- **memmgr** The `PODSMemoryMgr` object.
- **ptr** [in] Pointer to the range of memory to free.

Returns

None

See also

"smMalloc( )" on page 100

smMalloc( )

Allocate a block of memory.

Interface

PODSMemoryMgr

IDL definition

any smMalloc(PODSUInt32 *size*);

JavaScript synopsis

Not applicable

C synopsis

void* **PODSsmMalloc(**
  PODSMemoryMgr* *memmgr*,
  PODSUInt32 *size*
);

Parameters

- **memmgr** The `PODSMemoryMgr` object.
- **size** [in] The number of bytes of memory to allocate.

Returns

None

See also

"smFree( )" on page 99

smMemCopy( )

Copy a block of memory to another memory location.

Interface

PODSMemoryMgr

IDL definition

```
PODSInt16 smMemCopy(
  any dest,
  any src,
  PODSUInt32 length
);
```

JavaScript synopsis

Not applicable

C synopsis

```
PODSInt16 PODSsmMemCopy(
  PODSMemoryMgr* memmgr,
  void* dest,
  void* src,
  PODSUInt32 length
);
```

Parameters

- **memmgr** The `PODSMemoryMgr` object.
- **dest** [in] Pointer to destination to copy to.
- **src** [in] Pointer to source to copy from.
- **length** [in] Number of bytes to copy.

Returns

On Palm OS, returns 0 if successful, -1 otherwise.

On other platforms, return value is undefined.

See also

"smDup( )" on page 98, "smStrDup( )" on page 101

smStrDup( )

Duplicates a string.

Interface

PODSMemoryMgr

IDL definition

PODSString smStrDup(PODSString *str*);

JavaScript synopsis

Not applicable

C synopsis

```
PODSString PODSsmStrDup(
  PODSMemoryMgr* memmgr,
  PODSString str
);
```

Parameters

- **memmgr** The `PODSMemoryMgr` object.
- **str** [in] String to duplicate.

Returns

Copy of specified string.

See also

"smDup( )" on page 98, "smMemCopy( )" on page 100

smWrite( )

Writes specified memory to another location.

Interface

PODSMemoryMgr

IDL definition

```
PODSErr smWrite(
  any dest,
  PODSUInt32 offset,
  any src,
  PODSUInt32 length
);
```

JavaScript synopsis

Not applicable

C synopsis

```
PODSErr PODSsmWrite(
  PODSMemoryMgr*,
  void* dest,
  PODSUInt32 offset,
  void* src,
  PODSUInt32 length
);
```

Parameters

- **memmgr** The `PODSMemoryMgr` object.

- **dest** [in] Pointer to destination to copy to.
- **offset** [in] Memory offset from destination.
- **src** [in] Pointer to source to copy from.
- **length** [in] Number of bytes to write.

Returns

On Palm OS, returns 0 if successful, -1 otherwise.

On other platforms, return value is undefined.

See also

"smMemCopy( )" on page 100

PODSArray object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSAvantGo` object's "createStdArray( )" on page 86; or passed to or from JavaScript
- **Available to:** C, JavaScript

The attributes and methods in the `PODSArray` interface allow you to manipulate an array that either is to be exposed to JavaScript or has been received from JavaScript. For more information on exporting a POD to JavaScript, see "Writing C Code for PODS" on page 13 and "Using PODS functions from JavaScript engine" [*M-Business Anywhere Application Developer Guide*].

When you pass a JavaScript array to a PODS method that takes a `PODSArray` type, the `PODSArray` object is a PODS object that wraps the JavaScript array. You have the option of modifying this returned JavaScript array, then returning back to JavaScript. Then the `PODSArray` object would be unwrapped, so that the (modified) JavaScript array would be available to JavaScript.

You create the array using the `PODSAvantGo` object's "createStdArray( )" on page 86.

After using `createStdArray( )`, you need to call the array setter to initialize the array. See "Indexed properties" [*M-Business Anywhere Application Developer Guide*].

> **Note**
> When a PODS array is passed to JavaScript, although it is technically a PODS object, it acts like an ordinaryJavaScript array. Thus you can manipulate it in JavaScript just like you would if it were created in JavaScript.

Table 8. Summary of PODSArray attributes and methods

Description	Attributes and methods
Accessing array length	"length" on page 105
Accessing array elements	"getElement( )" on page 104 "setElement( )" on page 106

getElement( )

Gets the array element for a specified index value.

Interface

PODSArray

IDL definition

```
void getElement(
  PODSUInt32 index,
  PODSVariant obj
);
```

JavaScript synopsis

array.**getElement(***index***)**

C synopsis

```
void PODSgetElement(
  PODSArray* array,
  PODSUInt32 index,
  PODSVariant* obj
);
```

Parameters

- **array** The `PODSArray` object.
- **index** [in] Index of element to return.
- **obj** [out, retval] The array element object returned.

Returns

None

See also

"setElement( )" on page 106

length

The length of the specified array.

Interface

PODSArray

IDL definition

```
attribute PODSUInt32 length;
```

JavaScript synopsis

array.**length()**

array.**length()** = *length*

C synopsis

```
PODSUInt32 PODSgetLength(PODSArray* array);

void PODSsetLength(
  PODSArray* array,
```



```
  PODSUInt32 length
);
```

Parameters

- **array** The `PODSArray` object.
- **length** [in] Length for the specified array.

Returns

Getter: Length of the specified array.

Setter: None

setElement( )

Sets the array element for a specified index value.

Interface

PODSArray

IDL definition

```
void setElement(
  PODSUInt32 index,
  PODSVariant obj
);
```

JavaScript synopsis

array.**setElement(***index***)**

C synopsis

```
void PODSsetElement(
  PODSArray* array,
  PODSUInt32 index,
  PODSVariant*obj
);
```

Parameters

- **array** The `PODSArray` object.
- **index** [in] Index of element to return.
- **obj** [in] The array element object returned.

Returns

None

See also

"getElement( )" on page 104

PODSPlatform object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSAvantGo` object's "platform" on page 91
- **Available to:** C only

The `PODSPlatform` object provides functions that convert date values between platform-specific data types and the cross-platform `PODSDate` type.

Table 9. Summary of PODSPlatform methods

Description	Attributes and methods
Platform-to-PODS conversion	"convertPlatformDateToPODSDate( )" on page 107
PODS-to-platform conversion	"convertPODSDateToPlatformDate( )" on page 108

convertPlatformDateToPODSDate( )

Converts a platform-specific date value to the `PODSDate` value.

Interface

PODSPlatform

IDL definition

PODSDate convertPlatformDateToPODSDate(PODSPlatformDate *date*);

JavaScript synopsis

Not applicable

C synopsis

PODSDate **PODSconvertPlatformDateToPODSDate(**
  PODSPlatform* *platformobj*,
  PODSPlatformDate *date*
);

Parameters

- **platformobj** The `PODSPlatform` object.
- **date** [in] The platform-specific date value to convert.

Returns

The `PODSDate` value for the specified Platform-specific date value.

See also

"convertPODSDateToPlatformDate( )" on page 108

convertPODSDateToPlatformDate( )

Converts a `PODSDate` value to the platform-specific date value.

Interface

PODSPlatform

IDL definition

PODSPlatformDate convertPODSDateToPlatformDate(PODSDate *date*);

JavaScript synopsis

Not applicable

C synopsis

PODSPlatformDate **PODSconvertPODSDateToPlatformDate(**
  PODSPlatform* *platformobj*,
  PODSDate *date*
);

Parameters

- **platformobj** The `PODSPlatform` object.
- **date** [in] The `PODSDate` value to convert.

Returns

Platform-specific date value for the specified `PODSDate` value.

See also

"convertPlatformDateToPODSDate( )" on page 107

CHAPTER 7

PODS DOM-related objects

Contents

ADOMDOMImplementation object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSWindow` object's "document" on page 242 or `PODSDocument` object's "dom" on page 173
- **Available to:** C, JavaScript

The attributes and methods in the `ADOMDOMImplementation` interface implement most of the Worldwide Web Consortium (W3C) document object model (DOM) level 1 spec.

Full documentation of that spec is available online. See "Document Object Model (DOM)" [*M-Business Anywhere Application Developer Guide*] for links.

For a list of the W3C attributes and methods which have not been implemented, see "DOM level 1" [*M-Business Anywhere Application Developer Guide*].

The supported attributes and methods from the W3C DOM spec have been implemented without functional changes from that spec. Therefore the W3C documentation for those attributes and methods is not duplicated in this guide.

The synopsis required to call the M-Business C macros that implement the W3C DOM attributes and methods does add something to the W3C synopsis. "Using the W3C documentation to write M-Business DOM C code" on page 110 lists M-Business DOM C code necessary to call each of the W3C attributes and methods.

The "M-Business extensions to W3C DOM level 1" on page 157 details the methods that have been added to the W3C DOM spec in the M-Business implementation.

Note
M-Business Client has an upper limit of 65,535 DOM objects per HTML page. This is the total count of tags plus text objects. Pairs of opening and closing tags count as one. This limit applies equally, whether DOM objects are created by HTML markup or DOM methods. The example below contains five DOM objects (bold tag, text, plain text, italic tag, text): Bold text, plain text,<I> italic text.</I>

Creating a listener for hardware events

To create a listener for hardware events, you first write the event listener code in C and use "addEventListener( )" on page 158 to add and configure the event listener. Then you use the `PODSEventMgr` object's "registerEventHandler( )" on page 323 to register the event listener.

Using the W3C documentation to write M-Business DOM C code

Although W3C DOM attributes and methods implemented by M-Business Anywhere follow exactly the W3C DOM spec, special synopsis is required to call the corresponding M-Business DOM macro in C. (The JavaScript synopsis is standard.) "The W3C DOM spec and corresponding M-Business DOM

calls" on page 111 lists the W3C attributes and methods and the code required to call the corresponding M-Business attribute or method from JavaScript and from C. The table is in simple alphabetical order by W3C interface, then by keyword for each attribute or method defined.

The W3C DOM spec and corresponding M-Business DOM calls

The table below lists interfaces, attributes, and methods from the W3C DOM spec, along with the corresponding M-Business DOM (ADOM) synopsis for JavaScript and C. The table is in alphabetical order by W3C interface name. Within each interface, attributes and methods are listed in alphabetical order by the W3C keyword.

Note
The JavaScript synopsis column in this table does not attempt to be exhaustive in detailing different JavaScript usages; it merely provides an example of one of the more common JavaScript usages.

Table 1. W3C DOM spec and corresponding M-Business DOM calls

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`Attr` interface		
`readonly attribute DOMString name;`	*attrobj*.**name**	`ADOMString ADOMgetName( ADOMAttr* attrobj);`
`readonly attribute boolean specified;`	*attrobj*.**specified**	`ADOMBoolean ADOMgetSpecified( ADOMAttr* attrobj);`

`attribute DOMString value;`	*attrobj*.**value**
	
	attrobj.**value =** *value*
		`ADOMString ADOMgetValue( ADOMAttr* attrobj);`
		
		`void ADOMsetValue( ADOMAttr* attrobj, ADOMString value);`

`CharacterData` interface		
`void appendData( in DOMString arg) raises( DOMException);`	*chardataobj*.**appendData(** *arg*)	`void ADOMappendData( ADOMCharacterdata* chardataobj, ADOMString arg);`

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute DOMString **data**; // raises(DOMException) on setting // raises(DOMException) on retrieval	*chardataobj*.**data** *chardataobj*.**data** = *data*	ADOMString **ADOMgetData**(ADOMCharacterdata* *chardataobj*); void **ADOMsetData**(ADOMCharacterdata* *chardataobj*, ADOMString *data*);
void **deleteData**(in unsigned long *offset*, in unsigned long *count*) raises(DOMException);	*chardataobj*.**deleteData**(*offset*, *count*)	void **ADOMdeleteData**(ADOMCharacterdata* *chardataobj*, ADOMUInt32 *offset*, ADOMUInt32 *count*);
void **insertData**(in unsigned long *offset*,in DOMString *arg*)raises(DOMException);	*chardataobj*.**insertData**(*offset*, *arg*)	void **ADOMinsertData**(ADOMCharacterdata* *chardataobj*, ADOMUInt32 *offset*, ADOMString *arg*);
readonly attribute unsigned long **length**;	*chardataobj*.**length**	ADOMUInt32 **ADOMgetLength**(ADOMCharacterdata* *chardataobj*);
void **replaceData**(in unsigned long *offset*, in unsigned long *count*, in DOMString *arg*) raises(DOMException);	*chardataobj*. **replaceData**(*offset*, *count*, *arg*)	void **ADOMreplaceData**(ADOMCharacterdata* *chardataobj*, ADOMUInt32 *offset*, ADOMUInt32 *count*, ADOMString *arg*);
DOMString **substringData**(in unsigned long *offset*, in unsigned long *count*) raises(DOMException);	*chardataobj*. **substringData**(*offset*, *count*)	ADOMString **ADOMsubstringData**(ADOMCharacterdata* *chardataobj*, ADOMUInt32 *offset*, ADOMUInt32 *count*);
Document interface		
Element **createElement**(in DOMString *tagName*) raises(DOMException);	*docobj*.**createElement**(*tagName*)	ADOMElement* **ADOMcreateElement**(ADOMHTMLDocument* *docobj*, ADOMString *tagName*)
Text **createTextNode**(in DOMString *data*);	*docobj*.**createTextNode**(*data*)	ADOMText* **ADOMcreateTextNode**(ADOMHTMLDocument* *docobj*, ADOMString *data*)

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`readonly attribute Element documentElement;`	*docobj*.**documentElement**	`ADOMElement* ADOMgetDocumentElement(ADOMHTMLDocument* docobj)`
`NodeList getElementsByTagName (in DOMString tagname);`	*docobj*.**getElementsByTagName**(*tagname*)	`static ADOMNodeList* ADOMgetElementsByTagName (ADOMHTMLDocument* docobj, ADOMString tagname)`
`readonly attribute DOMImplementation implementation;`	*docobj*.**implementation**	`DOMImplementation* ADOMgetImplementation(ADOMHTMLDocument* docobj);`
`DOMImplementation` interface		
`boolean hasFeature( in DOMString feature,in DOMString version);`	*implemobj*.**hasFeature**(*feature*, *version*)	`ADOMBoolean ADOMhasFeature(ADOMImplementation* implemobj, ADOMString feature, ADOMString version);`
`Element` interface		
`DOMString getAttribute( in DOMString name);`	*elemobj*.**getAttribute**(*name*)	`ADOMString ADOMgetAttribute(ADOMElement* elemobj, ADOMString name);`
`Attr getAttributeNode( in DOMString name);`	*elemobj*.**getAttributeNode**(*name*)	`ADOMAttr* ADOMgetAttributeNode(ADOMElement* elemobj, ADOMString name);`
`NodeList getElementsByTagName (in DOMString tagname);`	*elemobj*.**getElementsByTagName** (*tagname*)	`static ADOMNodeList* ADOMgetElementsByTagName (ADOMHTMLDocument* docobj, ADOMString tagname)`
`void normalize();`	*elemobj*.**normalize**()	`void ADOMnormalize(ADOMElement* elemobj);`
`void removeAttribute( in DOMString name) raises( DOMException);`	*elemobj*.**removeAttribute**(*name*)	`void ADOMremoveAttribute(ADOMElement* elemobj, ADOMString name);`

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`Attr removeAttributeNode (in Attr oldAttr) raises( DOMException);`	`elemobj. removeAttributeNode( oldAttr)`	`ADOMAttr* ADOMremoveAttributeNode( ADOMElement* elemobj, Attr* oldAttr);`
`void setAttribute( in DOMString name, in DOMString value) raises( DOMException);`	`elemobj. setAttribute( name,value)`	`void ADOMsetAttribut( ADOMElement* elemobj, ADOMString name, ADOMString value);`
`Attr setAttributeNode( in Attr newAttr) raises( DOMException);`	`elemobj. setAttributeNode( newAttr)`	`ADOMAttr* ADOMsetAttributeNod( ADOMElement* elemobj, Attr* newAttr);`
`readonly attribute DOMString tagName;`	`elemobj.tagName`	`ADOMString ADOMgetTagName( ADOMElement* elemobj);`
`HTMLAnchorElement` interface		

`attribute DOMString accessKey;`	`anchorelemobj.accessKey`
	
	`anchorelemobj. accessKey = string`
		`ADOMString ADOMgetAccessKey( ADOMHTMLAnchorElement* anchorelemobj);`
		
		`void ADOMsetAccessKey( ADOMHTMLAnchorElement* anchorelemobj, ADOMString string);`

`void blur();`	`anchorelemobj.blur()`	`void ADOMblur( ADOMHTMLAnchorElement* anchorelemobj);`

`attribute DOMString charset;`	`anchorelemobj.charset`
	
	`anchorelemobj.charset = string`
		`ADOMString ADOMgetCharset( ADOMHTMLAnchorElement* anchorelemobj);`
		
		`void ADOMsetCharset( ADOMHTMLAnchorElement* anchorelemobj, ADOMString string);`

`attribute DOMString coords;`	`anchorelemobj.coords`
	
	`anchorelemobj.coords = string`
		`ADOMString ADOMgetCoords( ADOMHTMLAnchorElement* anchorelemobj);`
		
		`void ADOMsetCoords( ADOMHTMLAnchorElement* anchorelemobj, ADOMString string);`

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`void focus();`	*anchorelemobj*.**focus**()	void **ADOMfocus**(ADOMHTMLAnchorElement* *anchorelemobj*);
attribute DOMString **href**;	*anchorelemobj*.**href** *anchorelemobj*.**href** = *string*	ADOMString **DOMgetHref**(ADOMHTMLAnchorElement* *anchorelemobj*); void **ADOMsetHref**(ADOMAnchorElement* *anchorelemobj*, ADOMString *string*);
attribute DOMString **hreflang**;	*anchorelemobj*.**hreflang** *anchorelemobj*. **hreflang** = *string*	ADOMString **ADOMgetHreflang**(ADOMHTMLAnchorElement* *anchorelemobj*); void **ADOMsetHreflang**(ADOMHTMLAnchorElement* *anchorelemobj*, ADOMString *string*);
attribute DOMString **name**;	*anchorelemobj*.**name** *anchorelemobj*.**name**= *string*	ADOMString **ADOMgetName**(ADOMHTMLAnchorElement* *anchorelemobj*); void **ADOMsetName**(ADOMHTMLAnchorElement* *anchorelemobj*, ADOMString *string*);
attribute DOMString **rel**;	*anchorelemobj*.**rel** *anchorelemobj*.**rel** = *string*	ADOMString **ADOMgetRel**(ADOMHTMLAnchorElement* *anchorelemobj*); void **ADOMsetRel**(ADOMHTMLAnchorElement* *anchorelemobj*, ADOMString *string*);
attribute DOMString **rev**;	*anchorelemobj*.**rev** *anchorelemobj*.**rev** = *string*	ADOMString **ADOMgetRev**(ADOMHTMLAnchorElement* *anchorelemobj*); void **ADOMsetRev**(ADOMHTMLAnchorElement* *anchorelemobj*, ADOMString *string*);
attribute DOMString **shape**;	*anchorelemobj*.**shape** *anchorelemobj*.**shape** = *string*	ADOMString **ADOMgetShape**(ADOMHTMLAnchorElement* *anchorelemobj*); void **ADOMsetShape**(ADOMHTMLAnchorElement* *anchorelemobj*, ADOMString *string*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis

`attribute long tabIndex;`	*anchorelemobj*`.tabIndex`
	
	anchorelemobj`.tabIndex = `*tabindex*
		`ADOMInt32 ADOMgetTabIndex(`
		`ADOMHTMLAnchorElement*`
		anchorelemobj`);`
		
		`void ADOMsetTabIndex(`
		`ADOMHTMLAnchorElement*`
		anchorelemobj`,`
		`ADOMInt32 `*tabindex*`);`

`attribute DOMString target;`	*anchorelemobj*`.target`
	
	anchorelemobj`.target = string`
		`ADOMString ADOMgetTarget(`
		`ADOMHTMLAnchorElement*`
		anchorelemobj`);`
		
		`void ADOMsetTarget(`
		`ADOMHTMLAnchorElement*`
		anchorelemobj`,`
		`ADOMString `*string*`);`

`attribute DOMString type;`	*anchorelemobj*`.type`
	
	anchorelemobj`.type = type`
		`ADOMString ADOMgetType(`
		`ADOMHTMLAnchorElement*`
		anchorelemobj`);`
		
		`void ADOMsetType(`
		`ADOMHTMLAnchorElement*`
		anchorelemobj`,`
		`ADOMString `*type*`);`

`HTMLAreaElement` interface		

`attribute DOMString accessKey;`	*areaelemobj*`.accessKey`
	
	areaelemobj`.accessKey = string`
		`ADOMString DOMgetAccessKey(`
		`ADOMHTMLAreaElement*`
		areaelemobj`);`
		
		`void ADOMsetAccessKey(`
		`ADOMHTMLAreaElement*`
		areaelemobj`,`
		`ADOMString `*string*`);`

`attribute DOMString alt;`	*areaelemobj*`.alt`
	
	areaelemobj`.alt = string`
		`ADOMString DOMgetAlt(`
		`ADOMHTMLAreaElement*`
		areaelemobj`);`
		
		`void ADOMsetAlt(`
		`ADOMHTMLAreaElement*`
		`areaelemobj,`
		`ADOMString `*string*`);`

`attribute DOMString coords;`	*areaelemobj*`.coords`
	
	areaelemobj`.coords = string`
		`ADOMString ADOMgetCoords(`
		`ADOMHTMLAreaElement*`
		areaelemobj`);`
		
		`void ADOMsetCoords(`
		`ADOMHTMLAreaElement*`
		`areaelemobj,`
		`ADOMString `*string*`);`

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute DOMString **href**;	*areaelemobj*.**href** *areaelemobj*.**href** = *string*	ADOMString **ADOMgetHref**(ADOMHTMLAreaElement* *areaelemobj*); void **ADOMsetHref**(ADOMHTMLAreaElement* *areaelemobj*, ADOMString *string*);
attribute boolean **noHref**;	*areaelemobj*.**noHref** *areaelemobj*.**noHref** = *bool*	ADOMBoolean **ADOMgetNoHref**(ADOMHTMLAreaElement* *areaelemobj*); void **ADOMsetNoHref**(ADOMHTMLAreaElement* *areaelemobj*, ADOMBoolean *bool*);
attribute DOMString **shape**;	*areaelemobj*.**shape** *areaelemobj*.**shape** = *string*	ADOMString **ADOMgetShape**(ADOMHTMLAreaElement* *areaelemobj*); void **ADOMsetShape**(ADOMHTMLAreaElement* *areaelemobj*, ADOMString *string*);
attribute long **tabIndex**;	*areaelemobj*.**tabIndex** *areaelemobj*.**tabIndex** = *tabindex*	ADOMInt32 **ADOMgetTabIndex**(ADOMHTMLAreaElement* *areaelemobj*); void **ADOMsetTabIndex**(ADOMDOMImplementation* *adomobj*, ADOMInt32 *tabindex*);
attribute DOMString **target**;	*areaelemobj*.**target** *areaelemobj*.**target** = *string*	ADOMString **ADOMgetTarget**(ADOMHTMLAreaElement* *areaelemobj*); void **ADOMsetTarget**(ADOMHTMLAreaElement* *areaelemobj*, ADOMString *string*);
`HTMLBodyElement` interface		
attribute DOMString **aLink**;	*bodyelemobj*.**aLink** *bodyelemobj*.**aLink** = *string*	ADOMString **ADOMgetALink**(ADOMHTMLBodyElement* *bodyelemobj*); void **ADOMsetALink**(ADOMHTMLBodyElement* *bodyelemobj*, ADOMString *string*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute DOMString **background**;	*bodyelemobj*.**background** *bodyelemobj*. **background** = *string*	ADOMString **ADOMgetBackground**(ADOMHTMLBodyElement* *bodyelemobj*); void **ADOMsetBackground**(ADOMHTMLBodyElement* *bodyelemobj*, ADOMString *string*);
attribute DOMString **bgColor**;	*bodyelemobj*.**bgColor** *bodyelemobj*.**bgColor** = *string*	ADOMString **ADOMgetBgColor**(ADOMHTMLBodyElement* *bodyelemobj*); void **ADOMsetBgColor**(ADOMHTMLBodyElement* *bodyelemobj*, ADOMString *string*);
attribute DOMString **link**;	*bodyelemobj*.**link** *bodyelemobj*.**link** = *string*	ADOMString **ADOMgetLink**(ADOMHTMLBodyElement* *bodyelemobj*); void **ADOMsetLink**(ADOMHTMLBodyElement* *bodyelemobj*, ADOMString *string*);
attribute DOMString **text**;	*bodyelemobj*.**text** *bodyelemobj*.**text** = *string*	ADOMString **ADOMgetText**(ADOMHTMLBodyElement* *bodyelemobj*); void **ADOMsetText**(ADOMHTMLBodyElement* *bodyelemobj*, ADOMString *string*);
attribute DOMString **vLink**;	*bodyelemobj*.**vLink** *bodyelemobj*.**vLink** = *string*	ADOMString **ADOMgetVLink**(ADOMHTMLBodyElement* *bodyelemobj*); void **ADOMsetVLink**(ADOMHTMLBodyElement* *bodyelemobj*, ADOMString *string*);
HTMLBRElement interface		
attribute DOMString **clear**;	*breelemobj*.**clear** *breelemobj*.**clear** = *string*	ADOMString **ADOMgetClear**(ADOMHTMLBRElement* *breelemobj*); void **ADOMsetClear**(ADOMHTMLBRElement* *breelemobj*, ADOMString *string*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`HTMLButtonElement` interface		
`attribute DOMString accessKey;`	*btnelemobj*.**accessKey** *btnelemobj*.**accessKey** = *string*	`ADOMString` **ADOMgetAccessKey**(`ADOMHTMLButtonElement*` *btnelemobj*); `void` **ADOMsetAccessKey**(`ADOMHTMLButtonElement*` *btnelemobj*, `ADOMString` *string*);
`attribute boolean disabled;`	*btnelemobj*.**disabled** *btnelemobj*.**disabled** = *bool*	`ADOMBoolean` **ADOMgetDisabled**(`ADOMHTMLButtonElement*` *btnelemobj*); `void ADOMsetDisabled(` `ADOMHTMLButtonElement*` *btnelemobj*, `ADOMBoolean` *bool*);
`readonly attribute HTMLFormElement form;`	*btnelemobj*.button.**form**	`ADOMHTMLFormElement*` **ADOMgetForm**(`ADOMHTMLFormElement*` *formelemobj*);
`attribute DOMString name;`	*btnelemobj*.**name** *btnelemobj*.**name** = *string*	`ADOMString` **ADOMgetName**(`ADOMHTMLButtonElement*` *btnelemobj*); `void` **ADOMsetName**(`ADOMHTMLButtonElement*` *btnelemobj*, `ADOMString` *string*);
`attribute long tabIndex;`	*btnelemobj*.**tabIndex** *bnelemobj*.**tabIndex** = *tabindex*	`ADOMInt32` **ADOMgetTabIndex**(`ADOMHTMLButtonElement*` *btnelemobj*); `void` **ADOMsetTabIndex**(`ADOMHTMLButtonElement*` *btnelemobj*, `ADOMInt32` *tabindex*);
`readonly attribute DOMString type;`	*btnelemobj*.**type**	`ADOMString` **ADOMgetType**(`ADOMHTMLButtonElement*` *btnelemobj*);
`attribute DOMString value;`	*btnelemobj*.**value** *btnelemobj*.**value** = *value*	`ADOMString` **ADOMgetValue**(`ADOMHTMLButtonElement*` *btnelemobj*); `void` **ADOMsetValue**(`ADOMHTMLButtonElement*` *btnelemobj*, `ADOMString` *value*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`HTMLCollection` interface		
`Node item(in unsigned long index);`	*collectobj*`.item(`*index*`)`	`ADOMNode* ADOMitem(ADOMHTMLCollection* collectobj, ADOMUInt32 index);`
`readonly attribute unsigned long length;`	*collectobj*`.length`	`ADOMUInt32 ADOMgetLength(ADOMHTMLCollection* collectobj);`
`Node namedItem(in DOMString name);`	*collectobj*`.namedItem(`*name*`)`	`ADOMNode* ADOMnamedItem(ADOMHTMLCollection* collectobj, ADOMString name);`
`HTMLDirectoryElement` interface		
`attribute boolean compact;`	*directoryobj*`.compact` *directoryobj*`.compact = `*bool*	`ADOMBoolean ADOMgetCompact(ADOMHTMLDirectoryElement* direlemobj);` `void ADOMsetCompact(ADOMHTMLDirectoryElement* direlemobj, ADOMBoolean bool);`
`HTMLDivElement` interface		
`attribute DOMString align;`	*divobj*`.align` *divobj*`.align = `*string*	`ADOMString ADOMgetAlign(ADOMDivElement* divelemobj);` `void ADOMsetAlign(ADOMDivElement* divelemobj, ADOMString string);`
`HTMLDListElement` interface		
`attribute boolean compact;`	*dliseobj*`.compact` *dliseobj*`.compact = `*bool*	`ADOMBoolean ADOMgetCompact(ADOMHTMLDListElement* dlistelemobj);` `void ADOMsetCompact(ADOMHTMLDListElement* dlistelemobj, ADOMBoolean bool);`
`HTMLDocument` interface		

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`readonly attribute HTMLCollection anchors;`	*docobj*.**anchors**	`ADOMHTMLCollection* ADOMgetAnchors( ADOMHTMLDocument* docobj);`
`readonly attribute HTMLCollection applets;` (always returns NULL)	*docobj*.**applets**	`ADOMHTMLCollection* ADOMgetApplets( ADOMHTMLDocument* docobj);`
`attribute HTMLElement body;` (implemented as readonly)	*docobj*.**body**	`ADOMHTMLElement* ADOMgetBody( ADOMHTMLDocument* docobj);`
`readonly attribute DOMString domain;`	*docobj*.**domain**	`ADOMString ADOMgetDomain( ADOMHTMLDocument* docobj);`
`readonly attribute HTMLCollection forms;`	*docobj*.**forms**	`ADOMHTMLCollection* ADOMgetForms( ADOMHTMLDocument* docobj);`
`Element getElementById( DOMString elementId);`	*docobj*.**getElementById**(*elementId*)	`ADOMElement* ADOMgetElementById( ADOMHTMLDocument* docobj, ADOMString elementId);`
`NodeList getElementsByName( DOMString elementName);`	*docobj*. **getElementsByName**(*elementName*)	`ADOMNodeList* ADOMgetElementsByName( ADOMHTMLDocument* docobj, ADOMString elementName);`
`readonly attribute HTMLCollection images;`	*docobj*.**images**	`ADOMHTMLCollection* ADOMgetImages( ADOMHTMLDocument* docobj);`
`readonly attribute HTMLCollection links;`	*docobj*.**links**	`ADOMHTMLCollection* ADOMgetLinks( ADOMHTMLDocument* docobj);`
`readonly attribute DOMString referrer;`	*docobj*.**referrer**	`ADOMString ADOMgetReferrer( ADOMHTMLDocument* docobj);`

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis

`attribute DOMString title;`	*docobj*.**title**
	
	docobj.**title** = *title*
		`ADOMString ADOMgetTitle( ADOMHTMLDocument* docobj);`
		
		`void ADOMsetTitle( ADOMHTMLDocument* docobj, ADOMString title);`

`readonly attribute DOMString URL;`	*docobj*.**URL**	`ADOMString ADOMgetURL( ADOMHTMLDocument* docobj);`
`HTMLElement` interface		

`attribute DOMString className;`	*htmlelemobj*.**className**
	
	htmlelemobj.
	className = *string*
		`ADOMString ADOMgetClassName( ADOMHTMLElement* htmlelemobj);`
		
		`void ADOMsetClassName( ADOMHTMLElement* htmlelemobj, ADOMString string);`

`attribute DOMString dir;`	*htmlelemobj*.**dir**
	
	htmlelemobj.**dir** = *string*
		`ADOMString ADOMgetDir( ADOMHTMLElement* htmlelemobj);`
		
		`void ADOMsetDir( ADOMHTMLElement* htmlelemobj, ADOMString string);`

`attribute DOMString id;`	*htmlelemobj*.**id**
	
	htmlelemobj.elemname.**id** = *string*
		`ADOMString DOMgetId( ADOMHTMLElement* htmlelemobj);`
		
		`void ADOMsetId( ADOMHTMLElement* htmlelemobj, ADOMString string);`

`attribute DOMString lang;`	*htmlelemobj*.**lang**
	
	htmlelemobj.**lang** = *string*
		`ADOMString ADOMgetLang( ADOMHTMLElement* htmlelemobj);`
		
		`void ADOMsetLang( ADOMHTMLElement* htmlelemobj, ADOMString string);`

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute DOMString **title**;	*htmlelemobj*.**title** *htmlelemobj*.**title** = *string*	ADOMString **ADOMgetTitle**(ADOMHTMLElement* *htmlelemobj*); void **ADOMsetTitle**(ADOMHTMLElement* *htmlelemobj*, ADOMString *string*);
HTMLFontElement interface		
attribute DOMString **color**;	*fontelemobj*.**color** *fontelemobj*.**color** = *string*	ADOMString **ADOMgetColor**(ADOMHTMLFontElement* *fontelemobj*); void **ADOMsetColor**(ADOMHTMLFontElement* *fontelemobj*, ADOMString *string*);
attribute DOMString **face**;	*fontelemobj*.**face** *fontelemobj*.**face** = *string*	ADOMString **ADOMgetFace**(ADOMHTMLFontElement* *fontelemobj*); void **ADOMsetFace**(ADOMHTMLFontElement* *fontelemobj*, ADOMString *string*);
attribute DOMString **size**;	*fontelemobj*.**size** *fontelemobj*.**size** = *string*	ADOMString **ADOMgetSize**(ADOMHTMLFontElement* *fontelemobj*); void **ADOMsetSize**(ADOMHTMLFontElement* *fontelemobj*, ADOMString *string*);
HTMLFormElement interface		
attribute DOMString **acceptCharset**;	*formobj*.**acceptCharset** *formobj*.**acceptCharset** = *string*	ADOMString **ADOMgetAcceptCharset**(ADOMHTMLFormElement* *formobj*); void **ADOMsetAcceptCharset**(ADOMHTMLFormElement* *formobj*, ADOMString *string*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute DOMString **action**;	*formobj*.**action** *formobj*.**action** = *string*	ADOMString **ADOMgetAction**(ADOMHTMLFontElement* *fontelemobj*); void **ADOMsetAction**(ADOMHTMLFontElement* *fontelemobj*, ADOMString *string*);
readonly attribute HTMLCollection **elements**;	*formobj*.**elements**	ADOMHTMLCollection* **ADOMgetElements**(ADOMHTMLFontElement* *fontelemobj*);
attribute DOMString **enctype**;	*formobj*.**enctype** *formobj*.**enctype** = *string*	ADOMString **ADOMgetEnctype**(ADOMHTMLFontElement* *fontelemobj*); void **ADOMsetEnctype**(ADOMHTMLFontElement* *fontelemobj*, ADOMString *string*);
readonly attribute long **length**;	*formobj*.**length**	ADOMUInt32 **ADOMgetLength**(ADOMHTMLFontElement* *fontelemobj*);
attribute DOMString **method**;	*formobj*.**method** *formobj*.**method** = *string*	ADOMString **ADOMgetMethod**(ADOMHTMLFontElement* *fontelemobj*); void **ADOMsetMethod**(ADOMHTMLFontElement* *fontelemobj*, ADOMString *string*);
attribute DOMString **name**;	*formobj*.**name** *formobj*.**name** = *string*	ADOMString **ADOMgetName**(ADOMHTMLFontElement* *fontelemobj*); void **ADOMsetName**(ADOMHTMLFontElement* *fontelemobj*, ADOMString *string*);
void **reset**();	*formobj*.**reset**()	void **ADOMreset**(ADOMHTMLFontElement* *fontelemobj*);
void **submit**();	*formobj*.**submit**()	void **ADOMsubmit**(ADOMHTMLFontElement* *fontelemobj*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis

`attribute DOMString target;`	*formobj*.**target**
	
	formobj.**target** = *string*
		`ADOMString ADOMgetTarget( ADOMHTMLFontElement* fontelemobj);`
		
		`void ADOMsetTarget( ADOMHTMLFontElement* fontelemobj, ADOMString string);`

`HTMLHeadElement` interface		

`attribute DOMString profile;`	*headobj*.**profile**
	
	headobj.**profile** = *string*
		`ADOMString ADOMgetProfile( ADOMHTMLHeadElement* headobj);`
		
		`void ADOMsetProfile( ADOMHTMLHeadElement* headobj, ADOMString string);`

`HTMLHeadingElement` interface		

`attribute DOMString align;`	*headingobj*.**align**
	
	headingobj.**align** = *string*
		`ADOMString ADOMgetAlign( ADOMHTMLHeadElement* headingobj);`
		
		`void ADOMsetAlign( ADOMHTMLHeadElement* headingobj, ADOMString string);`

`HTMLHRElement` interface		

`attribute DOMString align;`	*hrobj*.**align**
	
	hrobj.**align** = *string*
		`ADOMString ADOMgetAlign( ADOMHTMLHRElement* hrobj);`
		
		`void ADOMsetAlign( ADOMHTMLHRElement* hrobj, ADOMString string);`

`attribute boolean noShade;`	*hrobj*.**noShade**
	
	hrobj.**noShade** = *bool*
		`ADOMBoolean ADOMgetNoShade( ADOMHTMLHRElement* hrobj);`
		
		`void ADOMsetNoShade( ADOMHTMLHRElement* hrobj, ADOMBoolean bool);`

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`attribute DOMString size;`	*hrobj*.**size** *hrobj*.**size** = *string*	ADOMString **ADOMgetSize**(ADOMHTMLHRElement* *hrobj*); void **ADOMsetSize**(ADOMHTMLHRElement* *hrobj*, ADOMString *string*);
`attribute DOMString width;`	*hrobj*.**width** *hrobj*.**width** = *string*	ADOMString **ADOMgetWidth**(ADOMHTMLHRElement* *hrobj*); void **ADOMsetWidth**(ADOMHTMLHRElement* *hrobj*, ADOMString *string*);
`HTMLHtmlElement` interface		
`attribute DOMString version;`	*htmlelemobj*.**version** *htmlelemobj*.**version** = *string*	ADOMString **ADOMgetVersion**(ADOMHTMLElementobject* *htmlelemobj*); void **ADOMsetVersion**(ADOMDOMImplementation* *htmlelemobj*, ADOMString *string*);
`HTMLImageElement` interface		
`attribute DOMString align;`	*imageelemobj*.**align** *imageelemobj*.**align** = *string*	ADOMString **ADOMgetAlign**(ADOMHTMLImageElement* *imageelemobj*); void **ADOMsetAlign**(ADOMHTMLImageElement* *imageelemobj*, ADOMString *string*);
`attribute DOMString alt;`	*imageelemobj*.**alt** *imageelemobj*.**alt** = *string*	ADOMString **ADOMgetAlt**(ADOMHTMLImageElement* *imageelemobj*); void **ADOMsetAlt**(ADOMHTMLImageElement* *imageelemobj*, ADOMString *string*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute DOMString **border**;	*imageelemobj*.**border** *imageelemobj*.**border** = *string*	ADOMString **ADOMgetBorder**(ADOMHTMLImageElement* *imageelemobj*); void **ADOMsetBorder**(ADOMHTMLImageElement* *imageelemobj*, ADOMString *string*);
attribute DOMString **height**;	*imageelemobj*.**height** *imageelemobj*.**height** = *string*	ADOMString **ADOMgetHeight**(ADOMHTMLImageElement* *imageelemobj*); void **ADOMsetHeight**(ADOMHTMLImageElement* *imageelemobj*, ADOMString *string*);
attribute DOMString **hspace**;	*imageelemobj*.**hspace** *imageelemobj*.**hspace** = *string*	ADOMString **ADOMgetHspace**(ADOMHTMLImageElement* *imageelemobj*); void **ADOMsetHspace**(ADOMHTMLImageElement* *imageelemobj*, ADOMString *string*);
attribute boolean **isMap**;	*imageelemobj*.**isMap** *imageelemobj*.**isMap** = *bool*	ADOMBoolean **ADOMgetIsMap**(ADOMHTMLImageElement* *imageelemobj*); void **ADOMsetIsMap**(ADOMHTMLImageElement* *imageelemobj*, ADOMBoolean *bool*);
attribute DOMString **longDesc**;	*imageelemobj*.**longDesc** *imageelemobj*.**longDesc** = *string*	ADOMString **ADOMgetLongDesc**(ADOMHTMLImageElement* *imageelemobj*); void **ADOMsetLongDesc**(ADOMHTMLImageElement* *imageelemobj*, ADOMString *string*);
attribute DOMString **lowSrc**;	*imageelemobj*.**lowSrc** *imageelemobj*.**lowSrc** = *string*	ADOMString **ADOMgetLowSrc**(ADOMHTMLImageElement* *imageelemobj*); void **ADOMsetLowSrc**(ADOMHTMLImageElement* *imageelemobj*, ADOMString *string*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`attribute DOMString name;`	*imageelemobj*.**name** *imageelemobj*.**name** = *string*	ADOMString **ADOMgetName**(ADOMHTMLImageElement* *imageelemobj*); void **ADOMsetName**(ADOMHTMLImageElement* *imageelemobj*, ADOMString *string*);
`attribute DOMString src;`	*imageelemobj*.**src** *imageelemobj*.**src** = *string*	ADOMString **ADOMgetSrc**(ADOMHTMLImageElement* *imageelemobj*); void **ADOMsetSrc**(ADOMHTMLImageElement* *imageelemobj*, ADOMString *string*);
`attribute DOMString useMap;`	*imageelemobj*.**useMap** *imageelemobj*.**useMap** = *string*	ADOMString **ADOMgetUseMap**(ADOMHTMLImageElement* *imageelemobj*); void **ADOMsetUseMap**(ADOMHTMLImageElement* *imageelemobj*, ADOMString *string*);
`attribute DOMString vspace;`	*imageelemobj*.**vspace** *imageelemobj*.**vspace** = *string*	ADOMString **ADOMgetVspace**(ADOMHTMLImageElement* *imageelemobj*); void **ADOMsetVspace**(ADOMHTMLImageElement* *imageelemobj*, ADOMString *string*);
`attribute DOMString width;`	*imageelemobj*.**width** *imageelemobj*.**width** = *string*	ADOMString **ADOMgetWidth**(ADOMHTMLImageElement* *imageelemobj*); void **ADOMsetWidth**(ADOMHTMLImageElement* *imageelemobj*, ADOMString *string*);
`HTMLInputElement` interface		
`attribute DOMString accept;`	*inputelemobj*.**accept** *inputelemobj*.**accept** = *string*	ADOMString **ADOMgetAccept**(ADOMHTMLInputElement* *inputelemobj*); void **ADOMsetAccept**(ADOMHTMLInputElement* *inputelemobj*, ADOMString *string*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute DOMString **accessKey**;	*inputelemobj*.**accessKey** *inputelemobj*. **accessKey** = *string*	ADOMString **ADOMgetAccessKey**(ADOMHTMLInputElement* *inputelemobj*); void **ADOMsetAccessKey**(ADOMHTMLInputElement* *inputelemobj*, ADOMString *string*);
attribute DOMString **align**;	*inputelemobj*.**align** *inputelemobj*.**align** = *string*	ADOMString **ADOMgetAlign**(ADOMHTMLInputElement* *inputelemobj*); void **ADOMsetAlign**(ADOMHTMLInputElement* *inputelemobj*, ADOMString *string*);
attribute DOMString **alt**;	*inputelemobj*.**alt** *inputelemobj*.**alt** = *string*	ADOMString **ADOMgetAlt**(ADOMHTMLInputElement* *inputelemobj*); void **ADOMsetAlt**(ADOMHTMLInputElement* *inputelemobj*, ADOMString *string*);
void **blur**();	*inputelemobj*.**blur**()	void **ADOMblur**(ADOMHTMLInputElement* *inputelemobj*);
attribute boolean **checked**;	*inputelemobj*.**checked** *inputelemobj*.**checked** = *bool*	ADOMBoolean **ADOMgetChecked**(ADOMHTMLInputElement* *inputelemobj*); void **ADOMsetChecked**(ADOMHTMLInputElement* *inputelemobj*, ADOMBoolean *bool*);
void **click**();	*inputelemobj*.**click**()	void **ADOMclick**(ADOMHTMLInputElement* *inputelemobj*);
attribute boolean **defaultChecked**;	*inputelemobj*. **defaultChecked** *inputelemobj*. **defaultChecked** = *bool*	ADOMBoolean **ADOMgetDefaultChecked**(ADOMHTMLInputElement* *inputelemobj*); void **ADOMsetDefaultChecked**(ADOMHTMLInputElement* *inputelemobj*, ADOMBoolean *bool*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute DOMString **defaultValue**;	*inputelemobj*.**defaultValue** *inputelemobj*.**defaultValue** = *string*	ADOMString **ADOMgetDefaultValue**(ADOMHTMLInputElement* *inputelemobj*); void **ADOMsetDefaultValue**(ADOMHTMLInputElement* *inputelemobj*, ADOMString *string*);
attribute boolean **disabled**;	*inputelemobj*.**disabled** *inputelemobj*.**disabled** = *bool*	ADOMBoolean **ADOMgetDisabled**(ADOMHTMLInputElement* *inputelemobj*); void **ADOMsetDisabled**(ADOMHTMLInputElement* *inputelemobj*, ADOMBoolean *bool*);
void **focus**();	*inputelemobj*.**focus**()	void **ADOMfocus**(ADOMHTMLInputElement* *inputelemobj*);
readonly attribute HTMLFormElement **form**;	*inputelemobj*.**form**	ADOMHTMLFormElement* **ADOMgetForm**(ADOMHTMLInputElement* *inputelemobj*);
attribute long **maxLength**;	*inputelemobj*.**maxLength** *inputelemobj*.**maxLength** = *maxlength*	ADOMInt32 **ADOMgetMaxLength**(ADOMHTMLInputElement* *inputelemobj*); void **ADOMsetMaxLength**(ADOMHTMLInputElement* *inputelemobj*, ADOMInt32 *maxlength*);
attribute DOMString **name**;	*inputelemobj*.**name** *inputelemobj*.**name** = string	ADOMString **ADOMgetName**(ADOMHTMLInputElement* *inputelemobj*); void **ADOMsetName**(ADOMHTMLInputElement* *inputelemobj*, ADOMString *string*);
attribute boolean **readOnly**;	*inputelemobj*.**readOnly** *inputelemobj*.**readOnly** = *bool*	ADOMBoolean **ADOMgetReadOnly**(ADOMHTMLInputElement* *inputelemobj*); void **ADOMsetReadOnly**(ADOMHTMLInputElement* *inputelemobj*, ADOMBoolean *bool*);
void **select**();	*inputelemobj*.**select**()	void **ADOMselect**(ADOMHTMLInputElement* *inputelemobj*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute DOMString **size**;	*inputelemobj*.**size** *inputelemobj*.**size** = *string*	ADOMString **ADOMgetSize**(ADOMHTMLInputElement* *inputelemobj*); void **ADOMsetSize**(ADOMHTMLInputElement* *inputelemobj*, ADOMString *string*);
attribute DOMString **src**;	*inputelemobj*.**src** *inputelemobj*.**src** = *string*	ADOMString **ADOMgetSrc**(ADOMHTMLInputElement* *inputelemobj*); void **ADOMsetSrc**(ADOMHTMLInputElement* *inputelemobj*, ADOMString *string*);
attribute long **tabIndex**;	*inputelemobj*.**tabIndex** *inputelemobj*.**tabIndex** = *tabindex*	ADOMInt32 **ADOMgetTabIndex**(ADOMHTMLInputElement* *inputelemobj*); void **ADOMsetTabIndex**(ADOMHTMLInputElement* *inputelemobj*, ADOMInt32 *tabindex*);
readonly attribute DOMString **type**;	*inputelemobj*.**type**	ADOMString **ADOMgetType**(ADOMHTMLInputElement* *inputelemobj*);
attribute DOMString **useMap**;	*inputelemobj*.**useMap** *inputelemobj*.**useMap** = *string*	ADOMString **ADOMgetUseMap**(ADOMHTMLInputElement* *inputelemobj*); void **ADOMsetUseMap**(ADOMHTMLInputElement* *inputelemobj*, ADOMString *string*);
attribute DOMString **value**;	*inputelemobj*.**value** *inputelemobj*.**value** = *value*	ADOMString **ADOMgetValue**(ADOMHTMLInputElement* *inputelemobj*); void **ADOMsetValue**(ADOMHTMLInputElement* *inputelemobj*, ADOMString *value*);
HTMLLIElement interface		

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis

`attribute DOMString type;`	`lielemobj.type`
	
	`lielemobj.type = type`
		`ADOMString ADOMgetType( ADOMHTMLIElement* lielemobj);`
		
		`void ADOMsetType( ADOMHTMLIElement* lielemobj, ADOMString type);`

`attribute long value;`	`lielemobj.value`
	
	`lielemobj.value = string`
		`ADOMString ADOMgetValue( ADOMHTMLIElement* lielemobj);`
		
		`void ADOMsetValue( ADOMHTMLIElement* lielemobj, ADOMString string);`

`HTMLMapElement` interface		
`readonly attribute HTMLCollection areas;`	`mapelemobj.areas`	`HTMLCollection* ADOMgetAreas( ADOMHTMLMapElement* mapelemobj);`

`attribute DOMString name;`	`mapelemobj.name`
	
	`mapelemobj.name = name`
		`ADOMString ADOMgetName( ADOMHTMLMapElement* mapelemobj);`
		
		`void ADOMsetName( ADOMHTMLMapElement* mapelemobj, ADOMString name);`

`HTMLMenuElement` interface		

`attribute boolean compact;`	`menuelemobj.compact`
	
	`menuelemobj.compact = bool`
		`ADOMBoolean ADOMgetCompact( ADOMHTMLMenuElement* menuelemobj);`
		
		`void ADOMsetCompact( ADOMHTMLMenuElement* menuelemobj, ADOMBoolean bool);`

`HTMLMetaElement` interface		

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute DOMString **content**;	*metaelemobj*.**content** *metaelemobj*.**content** = *string*	ADOMString **ADOMgetContent**(ADOMHTMLMetaElement* *metaelemobj*); void **ADOMsetContent**(ADOMHTMLMetaElement* *metaelemobj*, ADOMString *string*);
attribute DOMString **httpEquiv**;	*metaelemobj*.**httpEquiv** *metaelemobj*.**httpEquiv** = *string*	ADOMString **ADOMgetHttpEquiv**(ADOMHTMLMetaElement* *metaelemobj*); void **ADOMsetHttpEquiv**(ADOMHTMLMetaElement* *metaelemobj*, ADOMString *string*);
attribute DOMString **name**;	*metaelemobj*.**name** *metaelemobj*.**name** = *string*	ADOMString **ADOMgetName**(ADOMHTMLMetaElement* *metaelemobj*); void **ADOMsetName**(ADOMHTMLMetaElement* *metaelemobj*, ADOMString *string*);
attribute DOMString **scheme**;	*metaelemobj*.**scheme** *metaelemobj*.**scheme** = *string*	ADOMString **ADOMgetScheme**(ADOMHTMLMetaElement* *metaelemobj*); void **ADOMsetScheme**(ADOMHTMLMetaElement* *metaelemobj*, ADOMString *string*);
HTMLModElement interface		
attribute DOMString **cite**;	*modelemobj.modelemname*.**cite** *modelemobj.modelemname*.**cite** = *cite*	ADOMString **ADOMgetCite**(ADOMHTMLModElement* *modelemobj*); void **ADOMsetCite**(ADOMHTMLModElement* *modelemobj*, ADOMString *cite*);
attribute DOMString **dateTime**;	*modelemobj.modelemname*.**dateTime** *modelemobj.modelemname*.**dateTime** = *dateTime*	ADOMString **ADOMgetDateTime**(ADOMHTMLModElement* *modelemobj*); void **ADOMsetDateTime**(ADOMHTMLModElement* *modelemobj*, ADOMString *dateTime*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`HTMLObjectElement` interface		
`attribute DOMString align;`	*objelemobj*.**align** *objelemobj*.**align** = *align*	`ADOMString` **ADOMgetAlign**(`ADOMHTMLObjectElement*` *objelemobj*); `void` **ADOMsetAlign**(`ADOMHTMLObjectElement*` *objelemobj*, `ADOMString` *align*);
`attribute DOMString archive;`	*objelemobj*.**archive** *objelemobj*.**archive** = *archive*	`ADOMString` **ADOMgetArchive**(`ADOMHTMLObjectElement*` *objelemobj*); `void` **ADOMsetArchive**(`ADOMHTMLObjectElement*` *objelemobj*, `ADOMString` *archive*);
`attribute DOMString border;`	*objelemobj*.**border** *objelemobj*.**border** = *border*	`ADOMString` **ADOMgetBorder**(`ADOMHTMLObjectElement*` *objelemobj*); `void` **ADOMsetBorder**(`ADOMHTMLObjectElement*` *objelemobj*, `ADOMString` *border*);
`attribute DOMString code;`	*objelemobj*.**code** *objelemobj*.**code** = *code*	`ADOMString` **ADOMgetCode**(`ADOMHTMLObjectElement*` *objelemobj*); `void` **ADOMsetCode**(`ADOMHTMLObjectElement*` *objelemobj*, `ADOMString` *code*);
`attribute DOMString codeBase;`	*objelemobj*.**codeBase** *objelemobj*.**codeBase** = *codeBase*	`ADOMString` **ADOMgetCodeBase**(`ADOMHTMLObjectElement*` *objelemobj*); `void` **ADOMsetCodeBase**(`ADOMHTMLObjectElement*` *objelemobj*, `ADOMString` *codeBase*);
`attribute DOMString codeType;`	*objelemobj*.**codeType** *objelemobj*.**codeType** = *codeType*	`ADOMString` **ADOMgetCodeType**(`ADOMHTMLObjectElement*` *objelemobj*); `void` **ADOMsetCodeType**(`ADOMHTMLObjectElement*` *objelemobj*, `ADOMString` *codeType*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute DOMString **data**;	*objelemobj*.**data** *objelemobj*.**data** = *data*	ADOMString **ADOMgetData**(ADOMHTMLObjectElement* *objelemobj*); void **ADOMsetData**(ADOMHTMLObjectElement* *objelemobj*, ADOMString *data*);
attribute boolean **declare**;	*objelemobj*.**declare** *objelemobj*.**declare** = *bool*	ADOMBoolean **ADOMgetDeclare**(ADOMHTMLObjectElement* *objelemobj*); void **ADOMsetDeclare**(ADOMHTMLObjectElement* *objelemobj*, ADOMBoolean *bool*);
readonly attribute HTMLFormElement **form**;	*objelemobj*.**form**	HTMLFormElement* ADOMgetForm(ADOMHTMLObjectElement* *form*);
attribute DOMString **height**;	*objelemobj*.**height** *objelemobj*.**height** = *height*	ADOMString **ADOMgetHeight**(ADOMHTMLObjectElement* *objelemobj*); void **ADOMsetHeight**(ADOMHTMLObjectElement* *objelemobj*, ADOMString *height*);
attribute DOMString **hspace**;	*objelemobj*.**hspace** *objelemobj*.**hspace** = *hspace*	ADOMString **ADOMgetHspace**(ADOMHTMLObjectElement* *objelemobj*); void **ADOMsetHspace**(ADOMHTMLObjectElement* *objelemobj*, ADOMString *hspace*);
attribute DOMString **name**;	*objelemobj*.**name** *objelemobj*.**name** = *name*	ADOMString **ADOMgetName**(ADOMHTMLObjectElement* *objelemobj*); void **ADOMsetName**(ADOMHTMLObjectElement* *objelemobj*, ADOMString *name*);
attribute DOMString **standby**;	*objelemobj*.**standby** *objelemobj*.**standby** = *standby*	ADOMString **ADOMgetStandby**(ADOMHTMLObjectElement* *objelemobj*); void **ADOMsetStandby**(ADOMHTMLObjectElement* *objelemobj*, ADOMString *standby*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`attribute long tabIndex;`	*objelemobj*.**tabIndex** *objelemobj*.**tabIndex** = *tabindex*	ADOMInt32 **ADOMgetTabIndex**(ADOMHTMLObjectElement* *objelemobj*); void **ADOMsetTabIndex**(ADOMHTMLObjectElement* *objelemobj*, ADOMInt32 *tabindex*);
`attribute DOMString type;`	*objelemobj*.**type** *objelemobj*.**type** = *type*	ADOMString **ADOMgetType**(ADOMHTMLObjectElement* *objelemobj*); void **ADOMsetType**(ADOMHTMLObjectElement* *objelemobj*, ADOMString *type*);
`attribute DOMString useMap;`	*objelemobj*.**useMap** *objelemobj*.**useMap** = *useMap*	ADOMString **ADOMgetUseMap**(ADOMHTMLObjectElement* *objelemobj*); void **ADOMsetUseMap**(ADOMHTMLObjectElement* *objelemobj*, ADOMString *useMap*);
`attribute DOMString vspace;`	*objelemobj*.**vspace** *objelemobj*.**vspace** = *vspace*	ADOMString **ADOMgetVspace**(ADOMHTMLObjectElement* *objelemobj*); void **ADOMsetVspace**(ADOMHTMLObjectElement* *objelemobj*, ADOMString *vspace*);
`attribute DOMString width;`	*objelemobj*.**width** *objelemobj*.**width** = *width*	ADOMString **ADOMgetWidth**(ADOMHTMLObjectElement* *objelemobj*); void **ADOMsetWidth**(ADOMHTMLObjectElement* *objelemobj*, ADOMString *width*);
`HTMLOListElement` interface		
`attribute boolean compact;`	*olistelemobj*.**compact** *olistelemobj*.**compact** = *bool*	ADOMBoolean **ADOMgetCompact**(ADOMHTMLOListElement* *olistelemobj*); void **ADOMsetCompact**(ADOMHTMLOListElement* *olistelemobj*, ADOMBoolean *bool*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute long **start**;	*olistelemobj*.**start** *olistelemobj*.**start** = *start*	ADOMInt32 **ADOMgetStart**(ADOMHTMLOListElement* *olistelemobj*); void **ADOMsetStart**(ADOMHTMLOListElement* *olistelemobj*, ADOMInt32 *start*);
attribute DOMString **type**;	*olistelemobj*.**type** *olistelemobj*.**type** = *string*	ADOMString **ADOMgetType**(ADOMHTMLOListElement* *olistelemobj*); void **ADOMsetType**(ADOMHTMLOListElement* *olistelemobj*, ADOMString *string*);
HTMLOptionElement interface		
attribute boolean **defaultSelected**;	*optionelemobj*. **defaultSelected** *optionelemobj*. **defaultSelected** = *bool*	ADOMBoolean **ADOMgetDefaultSelected**(ADOMHTMLOptionElement* *optionelemobj*); void **ADOMsetDefaultSelected**(ADOMHTMLOptionElement* *optionelemobj*, ADOMBoolean *bool*);
attribute boolean **disabled**;	*optionelemobj*.**disabled** *optionelemobj*.**disabled** = *bool*	ADOMBoolean **ADOMgetDisabled**(ADOMHTMLOptionElement* *optionelemobj*); void **ADOMsetDisabled**(ADOMHTMLOptionElement* *optionelemobj*, ADOMBoolean *bool*);
readonly attribute HTMLFormElement **form**;	*optionelemobj*.**form**	HTMLFormElement* **ADOMgetForm**(ADOMHTMLOptionElement* *optionelemobj*);
readonly attribute long **index**;	*optionelemobj*.**index**	ADOMInt32 **ADOMgetIndex**(ADOMHTMLOptionElement* *optionelemobj*);
attribute DOMString **label**;	*optionelemobj*.**label** *optionelemobj*.**label** = *string*	ADOMString **ADOMgetLabel**(ADOMHTMLOptionElement* *optionelemobj*); void **ADOMsetLabel**(ADOMHTMLOptionElement* *optionelemobj*, ADOMString *string*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`readonly attribute boolean selected;`	*optionelemobj*.**selected**	`ADOMBoolean ADOMgetSelected( ADOMHTMLOptionElement* optionelemobj);`
`readonly attribute DOMString text;`	*optionelemobj*.**text**	`ADOMString ADOMgetText( ADOMHTMLOptionElement* optionelemobj);`

`attribute DOMString value;`	*optionelemobj*.**value**
	
	optionelemobj.**value** = *string*
		`ADOMString ADOMgetValue( ADOMHTMLOptionElement* optionelemobj);`
		
		`void ADOMsetValue( ADOMHTMLOptionElement* optionelemobj, ADOMString string);`

`HTMLParagraphElement` interface		

`attribute DOMString align;`	*pgfelemobj*.**align**
	
	pgfelemobj.**align** = *align*
		`ADOMString ADOMgetAlign( ADOMHTMLParagraphElement* pgfelemobj);`
		
		`void ADOMsetAlign( ADOMHTMLParagraphElement* pgfelemobj, ADOMString align);`

`HTMLParamElement` interface		

`attribute DOMString name;`	*paramelemobj*.**name**
	
	paramelemobj.**name** = *name*
		`ADOMString ADOMgetName( ADOMHTMLParamElement* paramelemobj);`
		
		`void ADOMsetName( ADOMHTMLParamElement* paramelemobj, ADOMString name);`

`attribute DOMString type;`	*paramelemobj*.**type**
	
	paramelemobj.**type** = *type*
		`ADOMString ADOMgetType( ADOMHTMLParamElement* paramelemobj);`
		
		`void ADOMsetType( ADOMHTMLParamElement* paramelemobj, ADOMString type);`

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis

`attribute DOMString value;`	*paramelemobj*`.value`
	
	paramelemobj`.value =` *value*
		`ADOMString ADOMgetValue(`
		`ADOMHTMLParamElement*` *paramelemobj*`);`
		
		`void ADOMsetValue(`
		`ADOMHTMLParamElement*` *paramelemobj*`,`
		`ADOMString` *value*`);`

`attribute DOMString valueType;`	*paramelemobj*`.valueType`
	
	paramelemobj`.valueType =` *valueType*
		`ADOMString ADOMgetValueType(`
		`ADOMHTMLParamElement*` *paramelemobj*`);`
		
		`void ADOMsetValueType(`
		`ADOMHTMLParamElement*` *paramelemobj*`,`
		`ADOMString` *valueType*`);`

`HTMLPreElement` interface		

`attribute long width;`	*preelemobj*`.width`
	
	preelemobj`.width =` *width*
		`ADOMString ADOMgetWidth(`
		`ADOMHTMLPreElement*` *preelemobj*`);`
		
		`void ADOMsetWidth(`
		`ADOMHTMLPreElement*` *preelemobj*`,`
		`ADOMString` *width*`);`

`HTMLQuoteElement` interface		

`attribute DOMString cite;`	*quoteelemobj*`.cite`
	
	quoteelemobj`.cite =` *cite*
		`ADOMString ADOMgetCite(`
		`ADOMHTMLQuoteElement*` *quoteelemobj*`);`
		
		`void ADOMsetCite(`
		`ADOMHTMLQuoteElement*` *quoteelemobj*`,`
		`ADOMString` *cite*`);`

`HTMLScriptElement` interface		

`attribute DOMString charset;`	*scriptelemobj*`.charset`
	
	scriptelemobj`.charset =` *charset*
		`ADOMString ADOMgetCharset(`
		`ADOMHTMLScriptElement*` *scriptelemobj*`);`
		
		`void ADOMsetCharset(`
		`ADOMHTMLScriptElement*` *scriptelemobj*`,`
		`ADOMString` *charset*`);`

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`attribute boolean defer;`	*scriptelemobj*.**defer** *scriptelemobj*.**defer** = *bool*	ADOMBoolean **ADOMgetDefer**(ADOMHTMLScriptElement* *scriptelemobj*); void **ADOMsetDefer**(ADOMHTMLScriptElement* *scriptelemobj*, ADOMBoolean *bool*);
`attribute DOMString event;`	*scriptelemobj*.**event** *scriptelemobj*.**event** = *event*	ADOMString **ADOMgetEvent**(ADOMHTMLScriptElement* *scriptelemobj*); void **ADOMsetEvent**(ADOMHTMLScriptElement* *scriptelemobj*, ADOMString *event*);
`attribute DOMString htmlFor;`	*scriptelemobj*.**htmlFor** *scriptelemobj*.**htmlFor** = *htmlFor*	ADOMString **ADOMgetHtmlFor**(ADOMHTMLScriptElement* *scriptelemobj*); void **ADOMsetHtmlFor**(ADOMHTMLScriptElement* *scriptelemobj*, ADOMString *htmlFor*);
`attribute DOMString src;`	*scriptelemobj*.**src** *scriptelemobj*.**src** = *src*	ADOMString **ADOMgetSrc**(ADOMHTMLScriptElement* *scriptelemobj*); void **ADOMsetSrc**(ADOMHTMLScriptElement* *scriptelemobj*, ADOMString *src*);
`attribute DOMString text;`	*scriptelemobj*.**text** *scriptelemobj*.**text** = *text*	ADOMString **ADOMgetText**(ADOMHTMLScriptElement* *scriptelemobj*); void **ADOMsetText**(ADOMHTMLScriptElement* *scriptelemobj*, ADOMString *text*);
`attribute DOMString type;`	*scriptelemobj*.**type** *scriptelemobj*.**type** = *type*	ADOMString **ADOMgetType**(ADOMHTMLScriptElement* *scriptelemobj*); void **ADOMsetType**(ADOMHTMLScriptElement* *scriptelemobj*, ADOMString *type*);
`HTMLSelectElement` interface		

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
void **add**(in HTMLElement element,in HTMLElement before);	*selectelemobj*.**add**(*element*, *before*)	void **ADOMadd**(ADOMHTMLSelectElement* *selectelemobj*, HTMLElement* *element*, HTMLElement* *before*);
void **blur**();	*selectelemobj*.**blur**()	void **ADOMblur**(ADOMHTMLSelectElement* *selectelemobj*);
attribute boolean **disabled**;	*selectelemobj*.**disabled** *selectelemobj*.**disabled** = *bool*	ADOMBoolean **ADOMgetDisabled**(ADOMHTMLSelectElement* *selectelemobj*); void **ADOMsetDisabled**(ADOMHTMLSelectElement* *selectelemobj*, ADOMBoolean *bool*);
void **focus**();	*selectelemobj*.**focus**()	void **ADOMfocus**(ADOMHTMLSelectElement* *selectelemobj*);
readonly attribute HTMLFormElement **form**;	*selectelemobj*.**form**	HTMLFormElement* **ADOMgetForm**(ADOMHTMLFormElement* *formmobj*);
readonly attribute long **length**;	*selectelemobj*.**length**	ADOMUInt32 **ADOMgetLength**(ADOMHTMLSelectElement* *selectelemobj*);
attribute boolean **multiple**;	*selectelemobj*.**multiple** *selectelemobj*.**multiple** = *bool*	ADOMBoolean **ADOMgetMultiple**(ADOMHTMLSelectElement* *selectelemobj*); void **ADOMsetMultiple**(ADOMHTMLSelectElement* *selectelemobj*, ADOMBoolean *bool*);
attribute DOMString **name**;	*selectelemobj*.**name** *selectelemobj*.**name** = **name**	ADOMString **ADOMgetName**(ADOMHTMLSelectElement* *selectelemobj*); void **ADOMsetName**(ADOMHTMLSelectElement* *selectelemobj*, ADOMString *name*);
readonly attribute HTMLCollection **options**;	*selectelemobj*.**options**	ADOMHTMLCollection* **ADOMgetOptions**(ADOMHTMLCollection* *collectobj*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`void remove( in long index);`	`selectelemobj. remove(index)`	`void ADOMremove( ADOMHTMLSelectElement* selectelemobj, ADOMInt32 index);`

`attribute long selectedIndex;`	`selectelemobj. selectedIndex`
	
	`selectelemobj. selectedIndex = selectedIndex`
		`ADOMInt32 ADOMgetSelectedIndex( ADOMHTMLSelectElement* selectelemobj);`
		
		`void ADOMsetSelectedIndex( ADOMHTMLSelectElement* selectelemobj, ADOMInt32 selectedIndex);`

`attribute long size;`	`selectelemobj.size`
	
	`selectelemobj.size = size`
		`ADOMString ADOMgetSize( ADOMHTMLSelectElement* selectelemobj);`
		
		`void ADOMsetSize( ADOMHTMLSelectElement* selectelemobj, ADOMString size);`

`attribute long tabIndex;`	`selectelemobj.tabIndex`
	
	`selectelemobj.tabIndex = tabindex`
		`ADOMInt32 ADOMgetTabIndex( ADOMHTMLSelectElement* selectelemobj);`
		
		`void ADOMsetTabIndex( ADOMHTMLSelectElement* selectelemobj, ADOMInt32 tabindex);`

`readonly attribute DOMString type;`	`selectelemobj.type`	`ADOMString ADOMgetType( ADOMHTMLSelectElement* selectelemobj);`

`attribute DOMString value;`	`selectelemobj.value`
	
	`selectelemobj.value = value`
		`ADOMString ADOMgetValue( ADOMHTMLSelectElement* selectelemobj);`
		
		`void ADOMsetValue( ADOMHTMLSelectElement* selectelemobj, ADOMString value);`

`HTMLStyleElement` interface		

`attribute boolean disabled;`	`styleelemobj.disabled`
	
	`styleelemobj.disabled = bool`
		`ADOMBoolean ADOMgetDisabled( ADOMHTMLStyleElement* styleelemobj);`
		
		`void ADOMsetDisabled( ADOMHTMLStyleElement* styleelemobj, ADOMBoolean bool);`

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`attribute DOMString media;`	*styleelemobj*.**media** *styleelemobj*.**media** = *media*	ADOMString **ADOMgetMedia**(ADOMHTMLStyleElement* *styleelemobj*); void **ADOMsetMedia**(ADOMHTMLStyleElement* *styleelemobj*, ADOMString *media*);
`attribute DOMString type;`	*styleelemobj*.**type** *styleelemobj*.**type** = *type*	ADOMString **ADOMgetType**(ADOMHTMLStyleElement* *styleelemobj*); void **ADOMsetType**(ADOMHTMLStyleElement* *styleelemobj*, ADOMString *type*);
`HTMLTableCaptionElement` interface		
`attribute DOMString align;`	*tblcapelemobj*.**align** *tblcapelemobj*.**align** = *align*	ADOMString **ADOMgetAlign**(ADOMHTMLTableCaption Element* *tblcapelemobj*); void **ADOMsetAlign**(ADOMHTMLTableCaption Element* *tblcapelemobj*, ADOMString *align*);
`HTMLTableCellElement` interface		
`attribute DOMString abbr;`	*tblcellelemobj*.**abbr** *tblcellelemobj*.**abbr** = *abbr*	ADOMString **ADOMgetAbbr**(ADOMHTMLTableCellElement* *tblcellelemobj*); void **ADOMsetAbbr**(ADOMHTMLTableCellElement* *tblcellelemobj*, ADOMString *abbr*);
`attribute DOMString align;`	*tblcellelemobj*.**align** *tblcellelemobj*.**align** = *align*	ADOMString **ADOMgetAlign**(ADOMHTMLTableCellElement* *tblcellelemobj*); void **ADOMsetAlign**(ADOMHTMLTableCellElement* *tblcellelemobj*, ADOMString *align*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute DOMString **axis**;	*tblcellelemobj*.**axis** *tblcellelemobj*.**axis** = *axis*	ADOMString **ADOMgetAxis**(ADOMHTMLTableCellElement* *tblcellelemobj*); void **ADOMsetAxis**(ADOMHTMLTableCellElement* *tblcellelemobj*, ADOMString *axis*);
attribute DOMString **bgColor**;	*tblcellelemobj*.**bgColor** *tblcellelemobj*.**bgColor** = *bgColor*	ADOMString **ADOMgetBgColor**(ADOMHTMLTableCellElement* *tblcellelemobj*); void **ADOMsetBgColor**(ADOMHTMLTableCellElement* *tblcellelemobj*, ADOMString *bgColor*);
readonly attribute long **cellIndex**;	*tblcellelemobj*. **cellIndex**	ADOMInt32 **ADOMgetCellIndex**(ADOMHTMLTableCellElement* *tblcellelemobj*);
attribute DOMString **ch**;	*tblcellelemobj*.**ch** *tblcellelemobj*.**ch** = *ch*	ADOMString **ADOMgetCh**(ADOMHTMLTableCellElement* *tblcellelemobj*); void **ADOMsetCh**(ADOMHTMLTableCellElement* *tblcellelemobj*, ADOMString *ch*);
attribute DOMString **chOff**;	*tblcellelemobj*.**chOff** *tblcellelemobj*.**chOff** = *chOff*	ADOMString **ADOMgetChOff**(ADOMHTMLTableCellElement* *tblcellelemobj*); void **ADOMsetChOff**(ADOMHTMLTableCellElement* *tblcellelemobj*, ADOMString *chOff*);
attribute long **colSpan**;	*tblcellelemobj*.**colSpan** *tblcellelemobj*.**colSpan** = *colSpan*	ADOMInt32 **ADOMgetColSpan**(ADOMHTMLTableCellElement* *tblcellelemobj*); void **ADOMsetColSpan**(ADOMHTMLTableCellElement* *tblcellelemobj*, ADOMInt32 *colSpan*);
attribute DOMString **headers**;	*tblcellelemobj*.**headers** *tblcellelemobj*.**headers** = *headers*	ADOMString **ADOMgetHeaders**(ADOMHTMLTableCellElement* *tblcellelemobj*); void **ADOMsetHeaders**(ADOMHTMLTableCellElement* *tblcellelemobj*, ADOMString *headers*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute DOMString **height**;	*tblcellelemobj*.**height** *tblcellelemobj*.**height** = *height*	ADOMString **ADOMgetHeight**(ADOMHTMLTableCellElement* *tblcellelemobj*); void **ADOMsetHeight**(ADOMHTMLTableCellElement* *tblcellelemobj*, ADOMString *height*);
attribute boolean **noWrap**;	*tblcellelemobj*.**noWrap** *tblcellelemobj*.**noWrap** = *bool*	ADOMBoolean **ADOMgetNoWrap**(ADOMHTMLTableCellElement* *tblcellelemobj*); void **ADOMsetNoWrap**(ADOMHTMLTableCellElement* *tblcellelemobj*, ADOMBoolean *bool*);
attribute long **rowSpan**;	*tblcellelemobj*.**rowSpan** *tblcellelemobj*.**rowSpan** = *rowSpan*	ADOMInt32 **ADOMgetRowSpan**(ADOMHTMLTableCellElement* *tblcellelemobj*); void **ADOMsetRowSpan**(ADOMHTMLTableCellElement* *tblcellelemobj*, ADOMInt32 *rowSpan*);
attribute DOMString **scope**;	*tblcellelemobj*.**scope** *tblcellelemobj*.**scope** = *scope*	ADOMString **ADOMgetScope**(ADOMHTMLTableCellElement* *tblcellelemobj*); void **ADOMsetScope**(ADOMHTMLTableCellElement* *tblcellelemobj*, ADOMString *scope*);
attribute DOMString **vAlign**;	*tblcellelemobj*.**vAlign** *tblcellelemobj*.**vAlign** = *vAlign*	ADOMString **ADOMgetVAlign**(ADOMHTMLTableCellElement* *tblcellelemobj*); void **ADOMsetVAlign**(ADOMHTMLTableCellElement* *tblcellelemobj*, ADOMString *vAlign*);
attribute DOMString **width**;	*tblcellelemobj*.**width** *tblcellelemobj*.**width** = *width*	ADOMString **ADOMgetWidth**(ADOMHTMLTableCellElement* *tblcellelemobj*); void **ADOMsetWidth**(ADOMHTMLTableCellElement* *tblcellelemobj*, ADOMString *width*);
HTMLTableColElement interface		

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis

`attribute DOMString align;`	*tblcolelemobj*.**align**
	
	tblcolelemobj.**align** = *string*
		`ADOMString ADOMgetAlign( ADOMHTMLTableColElement* tblcolelemobj);`
		
		`void ADOMsetAlign( ADOMHTMLTableColElement* tblcolelemobj, ADOMString string);`

`attribute DOMString ch;`	*tblcolelemobj*.**ch**
	
	tblcolelemobj.**ch** = *string*
		`ADOMString ADOMgetCh( ADOMHTMLTableColElement* tblcolelemobj);`
		
		`void ADOMsetCh( ADOMHTMLTableColElement* tblcolelemobj, ADOMString string);`

`attribute DOMString chOff;`	*tblcolelemobj*.**chOff**
	
	tblcolelemobj.**chOff** = *string*
		`ADOMString ADOMgetChOff( ADOMHTMLTableColElement* tblcolelemobj);`
		
		`void ADOMsetChOff( ADOMHTMLTableColElement* tblcolelemobj, ADOMString string);`

`attribute long span;`	*tblcolelemobj*.**span**
	
	tblcolelemobj.**span** = *span*
		`ADOMInt32 ADOMgetSpan( ADOMHTMLTableColElement* tblcolelemobj);`
		
		`void ADOMsetSpan( ADOMHTMLTableColElement* tblcolelemobj, ADOMInt32 span);`

`attribute DOMString vAlign;`	*tblcolelemobj*.**vAlign**
	
	tblcolelemobj.**vAlign** = *string*
		`ADOMString ADOMgetVAlign( ADOMHTMLTableColElement* tblcolelemobj);`
		
		`void ADOMsetVAlign( ADOMHTMLTableColElement* tblcolelemobj, ADOMString string);`

`attribute DOMString width;`	*tblcolelemname*.**width**
	
	tblcolelemname.**width** = *string*
		`ADOMString ADOMgetWidth( ADOMHTMLTableColElement* tblcolelemobj);`
		
		`void ADOMsetWidth( ADOMHTMLTableColElement* tblcolelemobj, ADOMString string);`

`HTMLTableElement` interface		

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute DOMString **align**;	*tblemobj*.**align** *tblemobj*.**align** = *string*	ADOMString **ADOMgetAlign**(ADOMHTMLTableElement* *tblemobj*); void **ADOMsetAlign**(ADOMHTMLTableElement* *tblemobj*, ADOMString *string*);
attribute DOMString **bgColor**;	*tblemobj*.**bgColor** *tblemobj*.**bgColor** = *string*	ADOMString **ADOMgetBgColor**(ADOMHTMLTableElement* *tblemobj*); void **ADOMsetBgColor**(ADOMHTMLTableElement* *tblemobj*, ADOMString *string*);
attribute DOMString **border**;	*tblemobj*.**border** *tblemobj*.**border** = *string*	ADOMString **ADOMgetBorder**(ADOMHTMLTableElement* *tblemobj*); void **ADOMsetBorder**(ADOMHTMLTableElement* *tblemobj*, ADOMString *string*);
attribute HTMLTableCaptionElement **caption**;	*tblemobj*.**caption** *tblemobj*.**caption** = *caption*	ADOMHTMLTableCaptionElement* **ADOMgetCaption**(ADOMHTMLTableElement* *tblemobj*); void **ADOMsetCaption**(ADOMHTMLTableElement* *tblemobj*, ADOMHTMLTableCaptionElement* *caption*);
attribute DOMString **cellPadding**;	*tblemobj*.**cellPadding** *tblemobj*.**cellPadding** = *string*	ADOMString **ADOMgetCellPadding**(ADOMHTMLTableElement* *tblemobj*); void **ADOMsetCellPadding**(ADOMHTMLTableElement* *tblemobj*, ADOMString *string*);
attribute DOMString **cellSpacing**;	*tblemobj*.**cellSpacing** *tblemobj*.**cellSpacing** = *string*	ADOMString **ADOMgetCellSpacing**(ADOMHTMLTableElement* *tblemobj*); void **ADOMsetCellSpacing**(ADOMHTMLTableElement* *tblemobj*, ADOMString *string*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`HTMLCaptionElement createCaption();`	`tblelemobj.createCaption()`	`ADOMHTMLCaptionElement* ADOMcreateCaption( ADOMHTMLTableElement* tblelemobj);`
`HTMLCaptionElement createTFoot();`	`tblelemobj.createTFoot()`	`ADOMHTMLCaptionElement* ADOMcreateTFoot( ADOMHTMLTableElement* tblelemobj);`
`HTMLCaptionElement createTHead();`	`tblelemobj.createTHead()`	`ADOMHTMLCaptionElement* ADOMcreateTHead( ADOMHTMLTableElement* tblelemobj);`
`void deleteCaption();`	`tblelemobj.deleteCaption()`	`void ADOMdeleteCaption( ADOMHTMLTableElement* tblelemobj);`
`void deleteRow( in long index);`	`tblelemobj.deleteRow(index)`	`void ADOMdeleteRow( ADOMHTMLTableElement* tblelemobj, ADOMInt32 index);`
`void deleteTFoot();`	`tblelemobj.deleteTFoot()`	`void ADOMdeleteTFoot( ADOMHTMLTableElement* tblelemobj);`
`void deleteTHead();`	`tblelemobj.deleteTHead()`	`void ADOMdeleteTHead( ADOMHTMLTableElement* tblelemobj);`

`attribute DOMString frame;`	`tblelemobj.frame`
	
	`tblelemobj.frame = string`
		`ADOMString ADOMgetFrame( ADOMHTMLTableElement* tblelemobj);`
		
		`void ADOMsetFrame( ADOMHTMLTableElement* tblelemobj, ADOMString string);`

`HTMLElement insertRow( in long index);`	`tblelemobj.insertRow(index)`	`ADOMHTMLElement* ADOMinsertRow( ADOMHTMLTableElement* tblelemobj, ADOMInt32 index);`
`readonly attribute HTMLCollection rows;`	`tblelemobj.rows`	`ADOMHTMLCollection* ADOMgetRows( ADOMHTMLTableElement* tblelemobj);`

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute DOMString **rules**;	*tblelemobj*.**rules** *tblelemobj*.**rules** = *string*	ADOMString **ADOMgetRules**(ADOMHTMLTableElement* *tblelemobj*); void **ADOMsetRules**(ADOMHTMLTableElement* *tblelemobj*, ADOMString *string*);
attribute DOMString **summary**;	*tblelemobj*.**summary** *tblelemobj*.**summary** = *string*	ADOMString **ADOMgetSummary**(ADOMHTMLTableElement* *tblelemobj*); void **ADOMsetSummary**(ADOMHTMLTableElement* *tblelemobj*, ADOMString *string*);
readonly attribute HTMLCollection **tBodies**;	*tblelemobj*.**tBodies**	ADOMHTMLCollection* **ADOMgetTBodies**(ADOMHTMLTableElement* *tblelemobj*);
attribute HTMLTableSection Element **tFoot**;	*tblelemobj*.**tFoot** *tblelemobj*.**tFoot** = *tfoot*	ADOMHTMLTableSectionElement* **ADOMgetTFoot**(ADOMHTMLTableElement* *tblelemobj*); void **ADOMsetTFoot**(ADOMHTMLTableElement* *tblelemobj*, HTMLTableSectionElement* *tfoot*);
attribute HTMLTableSection Element **tHead**;	*tblelemobj*.**tHead** *tblelemobj*.**tHead** = *thead*	ADOMHTMLTableSectionElement* **ADOMgetTHead**(ADOMHTMLTableElement* *tblelemobj*); void **ADOMsetTHead**(ADOMHTMLTableElement* *tblelemobj*, HTMLTableSectionElement* *thead*);
attribute DOMString **width**;	*tblelemobj*.**width** *tblelemobj*.**width** = *string*	ADOMString **ADOMgetWidth**(ADOMHTMLTableElement* *tblelemobj*); void **ADOMsetWidth**(ADOMHTMLTableElement* *tblelemobj*, ADOMString *string*);
HTMLTableRowElement interface		

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute DOMString **align**;	*tblrowelemobj*.**align** *tblrowelemobj*.**align** = *string*	ADOMString **ADOMgetAlign**(ADOMHTMLTableRowElement* *tblrowelemobj*); void **ADOMsetAlign**(ADOMHTMLTableRowElement* *tblrowelemobj*, ADOMString *string*);
attribute DOMString **bgColor**;	*tblrowelemobj*.**bgColor** *tblrowelemobj*.**bgColor** = *string*	ADOMString **ADOMgetBgColor**(ADOMHTMLTableRowElement* *tblrowelemobj*); void **ADOMsetBgColor**(ADOMHTMLTableRowElement* *tblrowelemobj*, ADOMString *string*);
readonly attribute HTMLCollection **cells**;	*tblrowelemobj*.**cells**	ADOMHTMLCollection* **ADOMgetCells**(ADOMHTMLTableRowElement* *tblrowelemobj*);
attribute DOMString **ch**;	*tblrowelemobj*.**ch** *tblrowelemobj*.**ch** = *string*	ADOMString **ADOMgetCh**(ADOMHTMLTableRowElement* *tblrowelemobj*); void **ADOMsetCh**(ADOMHTMLTableRowElement* *tblrowelemobj*, ADOMString *string*);
attribute DOMString **chOff**;	*tblrowelemobj*.**chOff** *tblrowelemobj*.**chOff** = *string*	ADOMString **ADOMgetChOff**(ADOMHTMLTableRowElement* *tblrowelemobj*); void **ADOMsetChOff**(ADOMHTMLTableRowElement* *tblrowelemobj*, ADOMString *string*);
void **deleteCell**(in long *index*);	*tblrowelemobj*. **deleteCell**(*index*)	void **ADOMdeleteCell**(ADOMHTMLTableRowElement* *tblrowelemobj*, ADOMInt32 *index*);
HTMLElement **insertCell**(in long *index*);	*tblrowelemobj*. **insertCell**(*index*)	ADOMHTMLElement* **ADOMinsertCell**(ADOMHTMLTableRowElement* *tblrowelemobj*, ADOMInt32 *index*);
readonly attribute long **rowIndex**;	*tblrowelemobj*.**rowIndex**	ADOMInt32 **ADOMgetRowIndex**(ADOMHTMLTableRowElement* *tblrowelemobj*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`readonly attribute long sectionRowIndex;`	*tblrowelemobj*`.sectionRowIndex`	`ADOMInt32 ADOMgetSectionRowIndex( ADOMHTMLTableRowElement* tblrowelemobj);`

`attribute DOMString vAlign;`	*tblrowelemobj*`.vAlign`
	
	tblrowelemobj`.vAlign =` *string*
		`ADOMString ADOMgetVAlign( ADOMHTMLTableRowElement* tblrowelemobj);`
		
		`void ADOMsetVAlign( ADOMHTMLTableRowElement* tblrowelemobj, ADOMString string);`

`HTMLTableSectionElement` interface		

`attribute DOMString align;`	*tblsectelemobj*`.align`
	
	tblsectelemobj`.align =` *string*
		`ADOMString ADOMgetAlign( ADOMHTMLTableSection Element* tblsectelemobj);`
		
		`void ADOMsetAlign( ADOMHTMLTableSection Element* tblsectelemobj, ADOMString string);`

`attribute DOMString ch;`	*tblsectelemobj*`.ch`
	
	tblsectelemobj`.ch =` *string*
		`ADOMString ADOMgetCh( ADOMHTMLTableSection Element* tblsectelemobj);`
		
		`void ADOMsetCh( ADOMHTMLTableSection Element* tblsectelemobj, ADOMString string);`

`attribute DOMString chOff;`	*tblsectelemobj*`.chOff`
	
	tblsectelemobj`.chOff =` *string*
		`ADOMString ADOMgetChOff( ADOMHTMLTableSection Element* tblsectelemobj);`
		
		`void ADOMsetChOff( ADOMHTMLTableSection Element* tblsectelemobj, ADOMString string);`

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`void deleteRow( in long index);`	`tblsectelemobj. deleteRow(index)`	`void ADOMdeleteRow( ADOMHTMLTableSection Element* tblsectelemobj, ADOMInt32 index);`
`HTMLElement insertRow(in long index);`	`tblsectelemobj. insertRow(index)`	`ADOMHTMLElement* ADOMinsertRow( ADOMHTMLTableSection Element* tblsectelemobj, ADOMInt32 index);`
`readonly attribute HTMLCollection rows;`	`tblsectelemobj.rows`	`ADOMHTMLCollection* ADOMgetRows( ADOMHTMLTableSection Element* tblsectelemobj);`

`attribute DOMString vAlign;`	`tblsectelemobj.vAlign`
	
	`tblsectelemobj.vAlign = string`
		`ADOMString ADOMgetVAlign( ADOMHTMLTableSection Element* tblsectelemobj);`
		
		`void ADOMsetVAlign( ADOMHTMLTableSection Element* tblsectelemobj, ADOMString string);`

`HTMLTextAreaElement` interface		

`attribute DOMString accessKey;`	`txtareaelemobj. accessKey`
	
	`txtareaelemobj. accessKey = string`
		`ADOMString ADOMgetAccessKey( ADOMHTMLTextAreaElement* txtareaelemobj);`
		
		`void ADOMsetAccessKey( ADOMHTMLTextAreaElement* txtareaelemobj, ADOMString string);`

`void blur();`	`txtareaelemobj.blur()`	`void ADOMblur( ADOMHTMLTextAreaElement* txtareaelemobj);`

`attribute long cols;`	`txtareaelemobj.cols`
	
	`txtareaelemobj.cols = cols`
		`ADOMInt32 ADOMgetColsAsInt( ADOMHTMLTextAreaElement* txtareaelemobj);`
		
		`void ADOMsetColsAsInt( ADOMHTMLTextAreaElement* txtareaelemobj, ADOMInt32 cols);`

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
attribute DOMString **defaultValue**;	*txtareaelemobj*. **defaultValue** *txtareaelemobj*. **defaultValue** = *string*	ADOMString **ADOMgetDefaultValue**(ADOMHTMLTextAreaElement* *txtareaelemobj*); void **ADOMsetDefaultValue**(ADOMHTMLTextAreaElement* *txtareaelemobj*, ADOMString *string*);
attribute boolean **disabled**;	*txtareaelemobj*.**disabled** *txtareaelemobj*.**disabled** = *bool*	ADOMBoolean **ADOMgetDisabled**(ADOMHTMLTextAreaElement* *txtareaelemobj*); void **ADOMsetDisabled**(ADOMHTMLTextAreaElement* *txtareaelemobj*, ADOMBoolean *bool*);
void **focus**();	*txtareaelemobj*.**focus**()	void **ADOMfocus**(ADOMHTMLTextAreaElement* *txtareaelemobj*);
readonly attribute HTMLFormElement **form**;	*txtareaelemobj*.**form**	ADOMHTMLFormElement* **ADOMgetForm**(ADOMHTMLFormElement* *formmobj*);
attribute DOMString **name**;	*txtareaelemobj*.**name** *txtareaelemobj*.**name** = *string*	ADOMString **ADOMgetName**(ADOMHTMLTextAreaElement* *txtareaelemobj*); void **ADOMsetName**(ADOMHTMLTextAreaElement* *txtareaelemobj*, ADOMString *string*);
attribute boolean **readOnly**;	*txtareaelemobj*.**readOnly** *txtareaelemobj*.**readOnly** = *bool*	ADOMBoolean **ADOMgetReadOnly**(ADOMHTMLTextAreaElement* *txtareaelemobj*); void **ADOMsetReadOnly**(ADOMHTMLTextAreaElement* *txtareaelemobj*, ADOMBoolean *bool*);
attribute long **rows**;	*txtareaelemobj*.**rows** *txtareaelemobj*.**rows** = *rows*	ADOMInt32 **ADOMgetRowsAsInt**(ADOMHTMLTextAreaElement* *txtareaelemobj*); void **ADOMsetRowsAsInt**(ADOMHTMLTextAreaElement* *txtareaelemobj*, ADOMInt32 *rows*);
void **select**();	*txtareaelemobj*.**select**()	void **ADOMselect**(ADOMHTMLTextAreaElement* *txtareaelemobj*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`attribute long tabIndex;`	*txtareaelemobj*.**tabIndex** *txtareaelemobj*.**tabIndex** = *tabindex*	ADOMInt32 **ADOMgetTabIndex**(ADOMHTMLTextAreaElement* *txtareaelemobj*); void **ADOMsetTabIndex**(ADOMHTMLTextAreaElement* *txtareaelemobj*, ADOMInt32 *tabindex*);
`readonly attribute DOMString type;`	*txtareaelemobj*.**type**	ADOMString **ADOMgetType**(ADOMHTMLTextAreaElement* *txtareaelemobj*);
`attribute DOMString value;`	*txtareaelemobj*.**value** *txtareaelemobj*.**value** = *string*	ADOMString **ADOMgetValue**(ADOMHTMLTextAreaElement* *txtareaelemobj*); void **ADOMsetValue**(ADOMHTMLTextAreaElement* *txtareaelemobj*, ADOMString *string*);
`HTMLTitleElement` interface		
`attribute DOMString text;`	*titleelemobj*.**text** *titleelemobj*.**text** = *string*	ADOMString **ADOMgetText**(ADOMHTMLTitleElement* *titleelemobj*); void **ADOMsetText**(ADOMHTMLTitleElement* *titleelemobj*, ADOMString *string*);
`HTMLUListElement` interface		
`attribute boolean compact;`	*ulistelemobj*.**compact** *ulistelemobj*.**compact** = *bool*	ADOMBoolean **ADOMgetCompact**(ADOMHTMLUListElement* *ulistelemobj*); void **ADOMsetCompact**(ADOMHTMLUListElement* *ulistelemobj*, ADOMBoolean *bool*);
`attribute DOMString type;`	*ulistelemobj*.**type** *ulistelemobj*.**type** = *string*	ADOMString **ADOMgetType**(ADOMHTMLUListElement* *ulistelemobj*); void **ADOMsetType**(ADOMHTMLUListElement* *ulistelemobj*, ADOMString *string*);
`Node` interface		

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
Node **appendChild**(in Node *newChild*) raises (DOMException);	*nodeobj*.**appendChild**(*newChild*)	ADOMNode* **ADOMappendChild**(ADOMNode* *nodeobj*, Node* *newChild*);
readonly attribute NamedNodeMap **attributes**;	*nodeobj*.**attributes**	ADOMNamedNodeMap* **ADOMgetAttributes**(ADOMNode* *nodeobj*);
readonly attribute NodeList **childNodes**;	*nodeobj*.**childNodes**	ADOMNodeList* **ADOMgetChildNodes**(ADOMNode* *nodeobj*);
Node **cloneNode**(in boolean *deep*);	*nodeobj*.**cloneNode**(*deep*)	ADOMNode* **ADOMcloneNode**(ADOMNode* *nodeobj*, ADOMBoolean *deep*);
readonly attribute Node **firstChild**;	*nodeobj*.**firstChild**	ADOMNode* **ADOMgetFirstChild**(ADOMNode* *nodeobj*);
boolean **hasChildNodes**();	*nodeobj*.**hasChildNodes**()	ADOMBoolean **ADOMhasChildNodes**(ADOMNode* *nodeobj*);
Node **insertBefore**(in Node *newChild*, in Node *refChild*) raises (DOMException);	*nodeobj*.**insertBefore**(*newChild*, *refChild*)	ADOMNode* **ADOMinsertBefore**(ADOMNode* *nodeobj*, ADOMNode* *newChild*, ADOMNode* *refChild*);
readonly attribute Node **lastChild**;	*nodeobj*.**lastChild**	ADOMNode* **ADOMgetLastChild**(ADOMNode* *nodeobj*);
readonly attribute Node **nextSibling**;	*nodeobj*.**nextSibling**	ADOMNode* **ADOMgetNextSibling**(ADOMNode* *nodeobj*);
readonly attribute DOMString **nodeName**;	*nodeobj*.**nodeName**	ADOMString **ADOMgetNodeName**(ADOMNode* *nodeobj*);
readonly attribute unsigned short **nodeType**;	*nodeobj*.**nodeType**	ADOMUInt16 **ADOMgetNodeType**(ADOMNode* *nodeobj*);
attribute DOMString **nodeValue**; // raises (DOMException) on setting // raises (DOMException) on retrieval	*nodeobj*.**nodeValue** *nodeobj*.**nodeValue** = *string*	ADOMString **ADOMgetNodeValue**(ADOMNode* *nodeobj*); void **ADOMsetNodeValue**(ADOMNode* *nodeobj*, ADOMString *string*);
readonly attribute Document **ownerDocument**;	*nodeobj*.**ownerDocument**	ADOMDocument* **ADOMgetOwnerDocument**(ADOMNode* *nodeobj*);

W3C DOM spec IDL definition	M-Business JavaScript synopsis	M-Business C synopsis
`readonly attribute Node parentNode;`	*nodeobj*.**parentNode**	`ADOMNode* ADOMgetParentNode( ADOMNode* nodeobj);`
`readonly attribute Node previousSibling;`	*nodeobj*.**previousSibling**	`ADOMNode* ADOMgetPreviousSibling( ADOMNode* nodeobj);`
`Node removeChild(in Node oldChild) raises (DOMException);`	*nodeobj*.**removeChild**(*oldChild*)	`ADOMNode* ADOMremoveChild( ADOMNode* nodeobj, ADOMNode* oldChild);`
`Node replaceChild( in Node newChild, in Node oldChild) raises (DOMException);`	*nodeobj*.**replaceChild**(*newChild*, *oldChild*)	`ADOMNode* ADOMreplaceChild( ADOMNode* nodeobj, ADOMNode* newChild, ADOMNode* oldChild);`
`NodeList` interface		
`Node item( in unsigned long index);`	*nodelistobj*.**item**(*index*)	`ADOMNode* ADOMitem( ADOMNodeList* nodelistobj, ADOMUInt32 index);`
`readonly attribute unsigned long length;`	*nodelistobj*.**length**	`ADOMUInt32 ADOMgetLength( ADOMNodeList* nodelistobj);`
`Text` interface		
`Text splitText( in unsigned long offset) raises (DOMException);`	*textobj*.**splitText**(*offset*)	`Text* ADOMsplitText( ADOMText* textobj, ADOMUInt32 offset);`

M-Business extensions to W3C DOM level 1

The `ADOMDDOMImplementation` is primarily an implementation of DOM level 1. M-Business Anywhere has selectively implemented a small number of DOM level 2 features, plus a few features that are not included in DOM level 2. All these extensions to DOM level 1 are documented in this section.

M-Business Anywhere extensions to the W3C DOM level 1 standard are listed below, grouped by function.

Table 2. M-Business Anywhere extensions to W3C DOM level 1, by function

Description	Attributes and methods
DOM level 2	"addEventListener( )" on page 158 "contentDocument" on page 159 "createDocument( )" on page 159 "createHTMLDocument( )" on page 160 "hasAttribute( )" on page 162 "normalize( )" on page 163 "ownerElement" on page 164 "removeEventListener( )" on page 164
Convenience	"getAttributeAsInt( )" on page 162 "getAttributeAsBoolean( )" on page 161 "setAttributeAsInt( )" on page 166 "setAttributeAsBoolean( )" on page 165

The M-Business Anywhere extensions to the W3C DOM level 1 standard are listed below, grouped by DOM object extended.

Table 3. M-Business Anywhere extensions to W3C DOM level 1, by DOM object

W3C DOM object	Attributes and methods
`Attr`	"ownerElement" on page 164
DOMImplementation	"createDocument( )" on page 159 "createHTMLDocument( )" on page 160

W3C DOM object	Attributes and methods
`Element`	"addEventListener( )" on page 158 "getAttributeAsBoolean( )" on page 161 "getAttributeAsInt( )" on page 162 "hasAttribute( )" on page 162 "removeEventListener( )" on page 164 "setAttributeAsBoolean( )" on page 165 "setAttributeAsInt( )" on page 166
HTMLObjectElement	"contentDocument" on page 159
`Node`	"normalize( )" on page 163

addEventListener( )

Adds an event listener for the specified event type. This is a DOM Level 2 method.

Interface

`ADOMHTMLElement` (W3C DOM: `EventTarget`)

IDL definition

```
void addEventListener(DOMString type,
  ADOMNativeScript cScript,
  any arg,
  boolean useCapture
);
```

JavaScript synopsis

htmlelem.**addEventListener(*type*, *cScript*, *useCapture*)**

C synopsis

```
void addEventListener(
  ADOMHTMLElement* htmlelem,
  ADOMString type,
  ADOMNativeScript* cScript,
  void* arg,
  PODSBoolean useCapture
);
```

Parameters

- ***htmlelem*** The `ADOMHTMLElement` object.
- ***type*** [in] The type of event; may be standard types, such as `onClick`, or user-defined types.

- ***cScript*** [in] Pointer to the listener script function.
- ***arg*** [in] An argument to pass to the listener script.
- ***useCapture*** [in] Not currently used; reserved for future implementation, so you must pass a boolean value here.

Returns

None

See also

"removeEventListener( )" on page 164

contentDocument

Gets the content document for the `ADOMHTMLObjectElement` object. This is a DOM Level 2 method.

Interface

`ADOMHTMLObjectElement`(W3C DOM: `HTMLObjectElement`)

IDL definition

readonly attribute Document contentDocument;

JavaScript synopsis

htmlobj.**contentDocument**

C synopsis

ADOMDocument* **ADOMgetContentDocument(**
  ADOMHTMLObjectElement* *htmlobj*
);

Parameters

- ***htmlobj*** The `ADOMHTMLObjectElement` object.

Returns

Content document. If you have a document within a document, in a frameset for example, then the content document is the inner document, which is contained by the outer document. This is a placeholder for future use.

See also

"createDocument( )" on page 159, "createHTMLDocument( )" on page 160

createDocument( )

Creates a document for a URL. This is a DOM Level 2 method.

Interface

ADOMDOMImplementation(W3C DOM: DOMImplementation)

IDL definition

HTMLDocument createDocument(DOMString *url*);

JavaScript synopsis

domimplobj.**CreateDocument(***url***)**

C synopsis

ADOMHTMLDocument* **ADOMcreateDocument(**
  ADOMHTMLDomImplementation* *domimplobj***,**
  ADOMString *url*
);

Parameters

- ***domimplobj*** The ADOMHTMLDomImplementation object.
- ***url*** [in] URL for document to be created.

Returns

Document created.

See also

"createHTMLDocument( )" on page 160

createHTMLDocument( )

Creates an HTML document. This is a DOM Level 2 method.

Interface

ADOMDOMImplementation(W3C DOM: DOMImplementation)

IDL definition

HTMLDocument createHTMLDocument(DOMString title);

JavaScript synopsis

domimplobj.**CreateHTMLDocument(***title***)**

C synopsis

ADOMHTMLDocument* **ADOMcreateHTMLDocument(**
  ADOMHTMLDomImplementation* *domimplobj***,**
  ADOMString *title*
);

Parameters

- ***domimplobj*** The ADOMHTMLDomImplementation object.

- ***title*** [in] The document title.

Returns

The document created.

See also

"createDocument( )" on page 159

getAttributeAsBoolean( )

Gets the boolean value for an `ADOMElement` object's named attribute.

Interface

`ADOMElement`(W3C DOM: `Element`)

IDL definition

boolean getAttributeAsBoolean(DOMString name);

JavaScript synopsis

elemobj.**GetAttributeAsBoolean(***name***)**

C synopsis

```
ADOMBoolean ADOMgetAttributeAsBoolean(
  ADOMElement* elemmobj,
  ADOMString name
);
```

Parameters

- ***elemobj*** The `ADOMElement` object.
- ***name*** [in] Name of attribute.

Returns

`PODS_TRUE`

If attribute is set to `PODS_TRUE`.

`PODS_FALSE`

Otherwise.

See also

"setAttributeAsBoolean( )" on page 165, "getAttributeAsInt( )" on page 162, "hasAttribute( )" on page 162

getAttributeAsInt( )

Gets the integer value for an ADOMElement's named attribute.

Interface

ADOMElement (W3C DOM: Element)

IDL definition

unsigned long getAttributeAsInt(in DOMString name);

JavaScript synopsis

elemobj.**GetAttributeAsInt(***name***)**

C synopsis

ADOMUInt32 **ADOMgetAttributeAsInt(**
ADOMElement* *elemobj***,**
ADOMString *name*
);

Parameters

- ***elemobj*** The ADOMElement object.
- ***name*** [in] Name of attribute.

Returns

Integer value of attribute.

See also

"setAttributeAsInt( )" on page 166, "getAttributeAsBoolean( )" on page 161, "hasAttribute( )" on page 162

hasAttribute( )

Determines if ADOMElement has named attribute. This is a DOM Level 2 method.

Interface

ADOMElement (W3C DOM: Element)

IDL definition

boolean hasAttribute(DOMString name);

JavaScript synopsis

document...*elemname*..**HasAttribute(***name***)**

C synopsis

ADOMBoolean **ADOMhasAttribute(**
ADOMElement* *elemobj***,**

ADOMString *name*
);

Parameters

- ***elemobj*** The `ADOMElement` object.
- ***name*** [in] Name of attribute.

Returns

`PODS_TRUE`

If `ADOMDOMImplementation` has the attribute.

`PODS_FALSE`

Otherwise.

Remarks

Results from `hasAttribute( )` will be unpredictable if the attribute was: (1) a `style` attribute property; and, (2) removed by the `removeAttribute( )` method of the DOM Element interface in "The W3C DOM spec and corresponding M-Business DOM calls" on page 111.

normalize( )

Normalizes an `ADOMNode` object. This is a DOM level 2 feature.

Interface

`ADOMNode` (W3C DOM: `Node`)

IDL definition

void normalize( );

JavaScript synopsis

nodeobj.**normalize( )**

C synopsis

void **ADOMnormalize(**ADOMNode* *nodeobj*);

Parameters

- ***nodeobj*** The `ADOMNode` object.

Returns

Normalized `ADOMNode` object.

Remarks

From the description in the W3C spec (http://www.w3.org/TR/DOM-Level-2-Core/core.html):

Puts all Text nodes in the full depth of the sub-tree underneath this Node, including attribute nodes, into a "normal" form where only structure (for example, elements, comments, processing instructions, CDATA sections, and entity references) separates Text nodes — that is, there are neither adjacent Text nodes nor empty Text nodes. This can be used to ensure that the DOM view of a document is the same as if it were saved and re-loaded, and is useful when operations (such as XPointer lookups) that depend on a particular document tree structure are to be used.

> In cases where the document contains CDATASections, the normalize operation alone may not be sufficient, since XPointers do not differentiate between Text nodes and CDATASection nodes.

ownerElement

Gets the owner element for an `ADOMAttr` object. This is a DOM Level 2 attribute.

Interface

`ADOMAttr` (W3C DOM: `Attr`)

IDL definition

readonly attribute Element ownerElement;

JavaScript synopsis

attrobj.**OwnerElement**

C synopsis

ADOMElement* **ADOMgetOwnerElement(**ADOMAttr* *attrobj***);**

Parameters

- ***attrobj*** The `ADOMAttr` object.

Returns

The owner element for the `ADOMAttr` object.

See also

Attr interface in "The W3C DOM spec and corresponding M-Business DOM calls" on page 111

removeEventListener( )

Removes the event listener for the specified event type. This is a DOM Level 2 method.

Interface

`ADOMHTMLElement` (W3C DOM: `EventTarget`)

IDL definition

```
void removeEventListener(
  DOMString type,
  ADOMNativeScript cScript,
  any arg,
  boolean useCapture
);
```

JavaScript synopsis

htmlelem.**removeEventListener(***type***,** *cScript***,** *arg***,** *useCapture***)**

C synopsis

```
void removeEventListener(
  ADOMHTMLElement* htmlelem,
  ADOMDOMString type,
  ADOMNativeScript* cScript,
  void* arg,
  ADOMBoolean useCapture
);
```

Parameters

- ***htmlelem*** The `ADOMHTMLElement` object.
- ***type*** [in] The type of event; may be standard types, such as `onClick` , or user-defined types.
- ***cScript*** [in] Pointer to the listener script function.
- ***arg*** [in] An argument to pass to the listener script.
- ***useCapture*** [in] Not currently used; reserved for future implementation, so you must pass a boolean value here.

Returns

None

See also

"addEventListener( )" on page 158

setAttributeAsBoolean( )

Sets a boolean value for a named attribute.

Interface

`ADOMElement` (W3C DOM: `Element`)

IDL definition

void setAttributeAsBoolean(in DOMString name, boolean)

JavaScript synopsis

elemobj.**SetAttributeAsBoolean(***name***,***boolean***)**

C synopsis

```
void ADOMsetAttributeAsBoolean(
  ADOMElement* elemobj,
  ADOMString name,
  ADOMBoolean boolean
);
```

Parameters

- ***elemobj*** The `ADOMElement` object.
- ***name*** [in] Name of attribute.
- ***boolean*** [in] Boolean value to set: `PODS_TRUE` or `PODS_FALSE`.

Returns

None

See also

"getAttributeAsBoolean( )" on page 161

setAttributeAsInt( )

Sets an integer value for a named attribute.

Interface

`ADOMElement` (W3C DOM: `Element`)

IDL definition

```
void setAttributeAsInt(
  in DOMString name,
  unsigned long int
);
```

JavaScript synopsis

elemobj.**SetAttributeAsInt(***name*, *int***)**

C synopsis

```
void ADOMsetAttributeAsInt(
  ADOMElement* elemobj,
  ADOMString name,
  ADOMUInt32 int
);
```

Parameters

- ***elemobj*** The `ADOMElement` object.
- ***name*** [in] Name of attribute.
- ***int*** [in] Integer value to set for attribute.

Returns

None

See also

“getAttributeAsInt( )” on page 162

Samples

This section lists code samples that illustrate the use of HTML tags listed earlier in this chapter.

Form action

```
<form action=http://search.yahoo.com/bin/search
   agsubmitMessage=Thanks
   agsubmitOffsiteLinks=true
   agsubmitIncludeImages=true
   agsubmitLinkDepth=2
   agsubmitSize=200
   agsubmitDiscardResponse=true
   agsubmitMultiple=false
   agsubmitHidden=false
   agsubmitDisplayDefaultMessage=true>

   <input size=15 name=p><BR>
   <INPUT TYPE=submit VALUE="Search">
   <INPUT TYPE=reset>
/form
```

CHAPTER 8

PODS document-related objects

Contents

PODSDocument object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSDocumentMgr` object's "createDocument( )" on page 187 (new document) or `PODSWindow` object's "document" on page 242 (document already displayed)
- **Available to:** C only

The `PODSDocument` object represents one document in the system, which can be an HTML page, an image, a script, or any one of several other objects. `PODSDocument` allows you to access all the document attributes, including the document source (`PODSDocumentSrc` object), and provides the document source data, a place where you can store information that you need to associate with the PODS document.

If you are vending a PODS document, you could first create it using the `PODSDocumentMgr` object's "createDocument( )" on page 187, and then use the attributes below to add to it.

If you are asking for a PODS document by URL, using the `PODSDocumentMgr` object's "documentForUrl( )" on page 183, you could use the attributes below to obtain information about the document.

Table 1. Summary of PODSDocument attributes and methods

Description	Attributes and methods
Accessing document source and source data	"documentSrc" on page 171 "documentSrcData" on page 172
Accessing general document information	"url" on page 178 "expirationDate" on page 174 "contentType" on page 170
Accessing information for particular document types	"title" on page 177 "titleCharset" on page 177 "dom" on page 173 "redirectUrl" on page 176 "imageWidth" on page 175 "imageHeight" on page 174

contentType

The document content type for the `PODSDocument` object.

Interface

`PODSDocument`

IDL definition

attribute PODSUInt8 contentType;

JavaScript synopsis

Not applicable

C synopsis

PODSUInt8 **PODSgetContentType(**PODSDocument* *doc*);

```
void PODSsetContentType(
  PODSDocument* doc,
  PODSUInt8 contype
);
```

Parameters

- ***doc*** The `PODSDocument` object.
- ***contype*** [in] Document content type, from the following list (defined in *pods.h* file): `PODS_REDIRECT_TYPE`, `PODS_UNKNOWN_TYPE`, `PODS_SCRIPT_TYPE`, `PODS_IMAGE_TYPE`, `PODS_HTML_TYPE`

Returns

Getter: Document content type, from list above.

Setter: None

Remarks

Use getter when asking for documents by URL, using the `PODSDocumentMgr` object's "documentForUrl( )" on page 183

See also

`PODSDocumentMgr` object's "documentForUrl( )" on page 183

documentSrc

The document source (the source that vended this document) for the `PODSDocument` object.

Interface

`PODSDocument`

IDL definition

attribute PODSDocumentSrc documentSrc;

JavaScript synopsis

Not applicable

C synopsis

PODSDocumentSrc* **PODSgetDocumentSrc(**PODSDocument* *doc*);

void **PODSsetDocumentSrc(**
PODSDocument* *doc*,
PODSDocumentSrc* *source*
);

Parameters

- ***doc*** The `PODSDocument` object.
- ***source*** [in] The document source to set.

Returns

Getter: The `PODSDocumentSrc` object or the `PODSDocument` object.

Setter: None

See also

"documentSrcData" on page 172

documentSrcData

The document source data for the `PODSDocument` object.

Interface

`PODSDocument`

IDL definition

attribute any documentSrcData;

JavaScript synopsis

Not applicable

C synopsis

void* **PODSgetDocumentSrcData(**PODSDocument* *doc*);

void **PODSsetDocumentSrcData(**
PODSDocument* *doc*,
void* *src*
);

Parameters

- ***doc*** The `PODSDocument` object.

- ***src*** [in] The document source data to set.

Returns

Getter: Pointer to document source data.

Setter: None

Remarks

The document source data is a memory area where you can store any private data that you need to have associated with the document. For example, you might want to store a pointer to a database handle so that you can close it later and free up the memory. You may structure the data you store in the document source data any way you like; it is only accessed by code that you write.

See also

"documentSrc" on page 171

dom

The document's `ADOMObject`.

Interface

`PODSDocument`

IDL definition

attribute ADOMHTMLDocument dom;

JavaScript synopsis

Not applicable

C synopsis

ADOMHTMLDocument* **PODSgetDom(**PODSDocument* *doc***)**

```
void PODSsetDom(
  PODSDocument* doc,
  ADOMHTMLDocument* dom
);
```

Parameters

- ***doc*** The `PODSDocument` object.
- ***dom*** [in] The document's `ADOMObject`.

Returns

Getter: The `ADOMObject` for the `PODSDocument` object.

Setter: None

Remarks

Getter used when you are vending a PODS document. The document `dom` is filled in if it is a `PODS_HTML_TYPE` document.

See also

"contentType" on page 170

expirationDate

The expiration date for the document.

Interface

`PODSDocument`

IDL definition

attribute PODSDate expirationDate;

JavaScript synopsis

Not applicable

C synopsis

PODSDate **PODSgetExpirationDate(**PODSDocument* *doc*);

void **PODSsetExpirationDate(**
PODSDocument* *doc*,
PODSDate *date*
);

Parameters

- ***doc*** The PODSDocument object.
- ***date*** [in] The expiration date to set.

Returns

Getter: The document's expiration date.

Setter: None

Remarks

If the document expiration date is set in the future, it is ignored by PODS. If it is set in the past, it is ignored by PODS for PODS documents, but for other documents PODS tries to get the document if the device is online.

imageHeight

The image height for the document; filled in if the document is a `PODS_IMAGE_TYPE` document.

Interface

PODSDocument

IDL definition

attribute PODSUInt16 imageHeight;

JavaScript synopsis

Not applicable

C synopsis

PODSUInt16 **PODSgetImageHeight(**PODSDocument* *doc*);

void **PODSsetImageHeight(**
PODSDocument* *doc*,
PODSUInt16 *height*
);

Parameters

- ***doc*** The PODSDocument object.
- ***height*** [in] The image height to set.

Returns

Getter: Document image height.

Setter: None

See also

"imageWidth" on page 175, "contentType" on page 170

imageWidth

The image width for the document; filled in if the document is a PODS_IMAGE_TYPE document.

Interface

PODSDocument

IDL definition

attribute PODSUInt16 imageWidth;

JavaScript synopsis

Not applicable

C synopsis

PODSUInt16 **PODSgetImageWidth(**PODSDocument* *doc*);

void **PODSsetImageWidth(**
PODSDocument* *doc*,


```
  PODSUInt16 width
);
```

Parameters

- ***doc*** The PODSDocument object.
- ***width*** [in] The image width to set.

Returns

Getter: Document image width.

Setter: None

See also

"imageHeight" on page 174, "contentType" on page 170

redirectUrl

The redirect URL for the document; filled in if the document is a `PODS_REDIRECT_TYPE` document.

Interface

`PODSDocument`

IDL definition

attribute PODSString redirectUrl;

JavaScript synopsis

Not applicable

C synopsis

PODSString **PODSgetRedirectUrl(**PODSDocument* *doc***);**

```
void PODSsetRedirectUrl(
  PODSDocument* doc,
  PODSString str
);
```

Parameters

- ***doc*** The PODSDocument object.
- ***str*** [in] The redirect URL to set.

Returns

Getter: Document redirect URL.

Setter: None

See also

"contentType" on page 170, "url" on page 178

title

The title for the document; filled in if the document is a `PODS_HTML_TYPE` document.

Interface

`PODSDocument`

IDL definition

attribute PODSString title;

JavaScript synopsis

Not applicable

C synopsis

PODSString **PODSgetTitle(**PODSDocument* *doc*);

void **PODSsetTitle(**
PODSDocument* *doc*,
PODSString *str*
);

Parameters

- ***doc*** The PODSDocument object.
- ***str*** [in] The title to set.

Returns

Getter: Document title.

Setter: None

titleCharset

The character set to be used to display the title of the `PODSDocument` object.

Interface

`PODSDocument`

IDL definition

attribute PODSUInt16 *titleCharset*;

JavaScript synopsis

Not applicable

C synopsis

PODSUInt16 **PODSgetTitleCharset(**PODSDocument* *doc***);**

void **PODSsetTitleCharset(**
PODSDocument* *doc***,**
PODSUInt16
);

Parameters

- ***doc*** The PODSDocument object.
- ***titleCharset*** [in] The character set to be used in displaying the document's title. For valid values, see "Constants to specify a title's character set" on page 60.

Returns

Getter: Value of character set to be used in displaying the document's title.

Setter: None

See also

"title" on page 215

url

The URL for the document.

Interface

`PODSDocument`

IDL definition

attribute PODSString url;

JavaScript synopsis

Not applicable

C synopsis

PODSString **PODSgetUrl(**PODSDocument* *doc***);**

void **PODSsetUrl(**
PODSDocument* *doc***,**
PODSString *str*
);

Parameters

- ***doc*** The PODSDocument object.
- ***str*** [in] The URL to set.

Returns

Getter: Document URL.

Setter: None

See also

"redirectUrl" on page 176

PODSDocumentSrc object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSDocument` object's "documentSrc" on page 171
- **Available to:** C only

`PODSDocumentSrc` is a single provider for documents, which are registered with the `PODSDocumentMgr`.

Most PODS authors will implement their own `PODSDocumentSrc` object and will implement either a "documentForUrl( )" on page 183 or a "documentForSubmission( )" on page 182, as well as a "closeDocument( )" on page 180. You need a "createDocumentEnumerator( )" on page 181 only if you want to vend documents and allow users to access them via the M-Business Client's Find.

Table 2. Summary of PODSDocumentSrc attributes and methods

Description	Attributes and methods
Setting up documents	"documentForUrl( )" on page 183 "documentForSubmission( )" on page 182
Closing or enumerating documents	"closeDocument( )" on page 180 "createDocumentEnumerator( )" on page 181

closeDocument( )

Closes the document.

Interface

`PODSDocumentSrc`

IDL definition

void closeDocument(PODSDocument *doc*);

JavaScript synopsis

Not applicable

C synopsis

```
void PODScloseDocument(
  PODSDocumentSrc* docsrc,
  PODSDocument* doc
);
```


Parameters

- ***docsrc*** The `PODSDocumentSrc` object.
- ***doc*** [in] The document to close.

Returns

None

Remarks

If you have used the shipped `PODSDocumentSrc` interface, when you call `destroy()` on a document, it automatically calls `closeDocument(°)` on the document's document source. If you implement your own `PODSDocumentSrc` interface, you also need to have your `destroy(°)` method automatically call `closeDocument(°)` on the document's document source.

createDocumentEnumerator( )

Creates a document enumerator.

Interface

`PODSDocumentSrc`

IDL definition

```
PODSDocumentEnumerator createDocumentEnumerator();
```

JavaScript synopsis

Not applicable

C synopsis

```
PODSDocumentEnumerator* PODScreateDocumentEnumerator(
  PODSDocumentSrc* docsrc
);
```

Parameters

- ***docsrc*** The `PODSDocumentSrc` object.

Returns

`PODSDocumentEnumerator` object for the `PODSDocumentSrc` object.

Remarks

Your document source can optionally implement this method, or your own implementation of it, if you would like it to provide enumeration of the documents that it can provide. Document enumeration allows the M-Business Client's Find to locate matching documents.

The `PODSDocumentSrc` implementation of createDocumentEnumerator( ) enumerates only a single document source, while the `PODSDocumentMgr` implementation enumerates all documents from all registered document sources.

See also

`PODSDocumentMgr` object's "createDocumentEnumerator( )" on page 188

documentForSubmission( )

Returns a document for submission.

Interface

`PODSDocumentSrc`

IDL definition

```
PODSDocument documentForSubmission(
  PODSSubmission submission,
  out PODSBoolean handled
);
```

JavaScript synopsis

Not applicable

C synopsis

```
PODSDocument* PODSdocumentForSubmission(
  PODSDocumentSrc* docsrc,
  PODSSubmission* submission,
  PODSBoolean* handled
);
```

Parameters

- ***docsrc*** The `PODSDocumentSrc` object.
- ***submission*** [in] The `PODSSubmission` object.
- ***handled*** [out] Indicates whether the request was handled successfully: `PODS_TRUE` on success; `PODS_FALSE` otherwise. You may return *handled* as `PODS_TRUE` while returning a null `PODSDocument`. For example, you might want to have a link with a URL that does not return a page, but toggles the device backlighting; the URL would be something like `"PODS://togglebacklight"`.

Returns

`PODSDocument` object for specified submission. May be `NULL`.

Remarks

Your document source should implement this method and/or `documentForUrl(°)`.

The `PODSDocumentSrc` implementation of `documentForSubmission(°)` only checks the specified document source, while the `PODSDocumentMgr` implementation walks through all the registered document sources until a document with a matching submission object is found.

See also

"documentForUrl( )" on page 183, PODSDocumentMgr object's
"documentForSubmission( )" on page 189

documentForUrl( )

Returns a document for a URL.

Interface

PODSDocumentSrc

IDL definition

```
PODSDocument documentForUrl(PODSString url,
  out PODSBoolean handled
);
```

JavaScript synopsis

Not applicable

C synopsis

```
PODSDocument* PODSdocumentForUrl(
  PODSDocumentSrc* docsrc,
  PODSString str,
  PODSBoolean* handled
);
```

Parameters

- ***docsrc*** The PODSDocumentSrc object.
- ***str*** [in] The URL for the document.
- ***handled*** [out] Indicates whether the request was handled successfully: PODS_TRUE on success; PODS_FALSE otherwise. You may return *handled* as PODS_TRUE while returning a NULL PODSDocument. Returning *handled* as PODS_TRUE merely indicates that no further processing is needed.

Returns

PODSDocument object for the specified. May be NULL.

Remarks

Your document source should implement this method and/or documentForSubmission(°).

The PODSDocumentSrc implementation of documentForUrl(°) only checks the specified document source, while the PODSDocumentMgr implementation walks through all the registered document sources until a document with a matching URL is found.

See also

"documentForSubmission( )" on page 182, `PODSDocumentMgr` object's
"documentForUrl( )" on page 190

PODSDocumentEnumerator object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSDocumentSrc` object's "createDocumentEnumerator( )" on page 181, or `PODSDocumentMgr` object's "createDocumentEnumerator( )" on page 188
- **Available to:** C only

The `PODSDocumentEnumerator`'s `nextDocument(°)` method allows you to access a series of documents in sequence. The `PODSDocumentEnumerator` object is created by the `PODSDocumentSrc` object's "createDocumentEnumerator( )" on page 181.

Table 3. Summary of PODSDocumentEnumerator attributes and methods

Description	Attributes and methods
Accessing the next document	"nextDocument( )" on page 185

nextDocument( )

Gets the next document in sequence.

Interface

`PODSDocumentEnumerator`

IDL definition

```
PODSDocument nextDocument( );
```

JavaScript synopsis

Not applicable

C synopsis

```
PODSDocument* PODSnextDocument(
  PODSDocumentEnumerator* docenum
);
```

Parameters

- ***docenum*** The `PODSDocumentEnumerator` object.

Returns

The next document in the `PODSDocumentEnumerator` sequence. `NULL` when there are no more documents.

Remarks

Do not access the prior document after calling `nextDocument(°)` because it may already have been destroyed. The final document is destroyed when the document enumerator is destroyed.

PODSDocumentMgr object

- **Inherits from:** `PODSDocumentSrc`
- **Accessed by:** `PODSAvantGo` object's "documentMgr" on page 87
- **Available to:** C only

The `PODSDocumentMgr` methods register, unregister, and create documents.

`PODSDocumentMgr`'s implementations of `documentForUrl(°)` and `documentForSubmission(°)` (inherited from `PODSDocumentSrc`) call the corresponding method on each of the document sources that have been registered with the document manager. If any of the document sources mark `handled` as `PODS_TRUE`, the search stops and if a document is returned from the document source, it is returned by the document manager. Note that you may set `handled` to `PODS_TRUE`, indicating that you have handled the request, but still return `NULL`.

`PODSDocumentMgr`'s implementation of `createDocumentEnumerator(°)` (inherited from `PODSDocumentSrc`) creates a document enumerator that returns each document that the document sources can provide. As each document source's enumerator is exhausted, the next document source is asked to create an enumerator until no more documents are returned.

Table 4. Summary of PODSDocumentMgr attributes and methods

Description	Attributes and methods
Registering/unregistering documents	"registerDocumentSrc( )" on page 191 "unregisterDocumentSrc( )" on page 192
Creating documents	"createDocument( )" on page 187
Enumerating or returning documents	"createDocumentEnumerator( )" on page 188 "documentForSubmission( )" on page 189 "documentForUrl( )" on page 190

createDocument( )

Creates a document.

Interface

`PODSDocumentMgr`

IDL definition

```
PODSDocument createDocument(
  PODSString url,
```



```
  PODSDocumentContentType type
);
```

JavaScript synopsis

Not applicable

C synopsis

```
PODSDocument* PODScreateDocument(
  PODSDocumentMgr* docmgr,
  PODSString url,
  PODSDocumentContentType type
);
```

Parameters

- ***docmgr*** The `PODSDocumentMgr` object.
- ***url*** [in] The URL for the document.
- ***type*** [in] Document content type, from the following list (defined in *pods.h* file): `PODS_REDIRECT_TYPE`, `PODS_UNKNOWN_TYPE`, `PODS_SCRIPT_TYPE`, `PODS_IMAGE_TYPE`, `PODS_HTML_TYPE`

Returns

The document created.

Remarks

Specifying `PODS_HTML_TYPE` for *type* automatically creates a document object, on which you can call the `PODSDocument` object's "dom" on page 173.

See also

`PODSDocument` object's "dom" on page 173

createDocumentEnumerator( )

Creates a document enumerator.

Interface

`PODSDocumentMgr`

IDL definition

```
PODSDocumentEnumerator createDocumentEnumerator( );
```

JavaScript synopsis

Not applicable

C synopsis

```
PODSDocumentEnumerator* PODScreateDocumentEnumerator(
  PODSDocumentMgr* docmgr
);
```

Parameters

- ***docmgr*** The `PODSDocumentMgr` object.

Returns

`PODSDocumentEnumerator` object for the `PODSDocumentMgr` object.

Remarks

Your document manager can optionally implement this method, or your own implementation of it, if you would like it to provide enumeration of the documents that it can provide. Document enumeration allows the M-Business Client's Find to locate matching documents.

The `PODSDocumentMgr` implementation of createDocumentEnumerator( ) enumerates all documents from all registered document sources, while the `PODSDocumentSrc` implementation enumerates only a single document source.

See also

`PODSDocumentSrc` object's "createDocumentEnumerator( )" on page 181

documentForSubmission( )

Returns a document for a submission.

Interface

`PODSDocumentMgr`

IDL definition

```
PODSDocument documentForSubmission(
  PODSSubmission submission,
  out PODSBoolean handled
);
```

JavaScript synopsis

Not applicable

C synopsis

```
PODSDocument* PODSdocumentForSubmission(
  PODSDocumentMgr* docmgr,
  PODSSubmission* submission,
  PODSBoolean* handled
);
```


Parameters

- ***docmgr*** The `PODSDocumentMgr` object.
- ***submission*** [in] The `PODSSubmission` object.
- ***handled*** [out] Indicates whether the request was handled successfully: `PODS_TRUE` on success; `PODS_FALSE` otherwise. You may return *handled* as `PODS_TRUE` while returning a null `PODSDocument`. For example, you might want to have a link with a URL that does not return a page, but toggles the device backlighting; the URL would be something like `"PODS://togglebacklight"`.

Returns

`PODSDocument` object for specified submission. May be `NULL`.

Remarks

Your document manager should implement this method and/or `documentForUrl(°)`.

The `PODSDocumentMgr` implementation of `documentForSubmission(°)` walks through all the registered document sources until a document with a matching submission object is found, while the `PODSDocumentSrc` implementation only checks the specified document source.

See also

"documentForUrl( )" on page 190, `PODSDocumentSrc` object's
"documentForSubmission( )" on page 182

documentForUrl( )

Returns a document for a URL.

Interface

`PODSDocumentMgr`

IDL definition

```
PODSDocument documentForUrl(
  PODSString url,
  out PODSBoolean handled
);
```

JavaScript synopsis

Not applicable

C synopsis

```
PODSDocument* PODSdocumentForUrl(
  PODSDocumentMgr* docmgr,
  PODSString str,
  PODSBoolean* handled
);
```


Parameters

- ***docmgr*** The `PODSDocumentMgr` object.
- ***str*** [in] The URL for the document.
- ***handled*** [out] Indicates whether the request was handled successfully: `PODS_TRUE` on success; `PODS_FALSE` otherwise. You may return *handled* as `PODS_TRUE` while returning a null `PODSDocument`. Returning *handled* as `PODS_TRUE` merely indicates that no further processing is needed.

Returns

`PODSDocument` object for specified URL. May be `NULL`.

Remarks

Your document manager should implement this method and/or `documentForSubmission(°)`.

The `PODSDocumentMgr` implementation of `documentForUrl(°)` walks through all the registered document sources until a document with a matching URL is found, while the `PODSDocumentSrc` implementation only checks the specified document source.

See also

"documentForSubmission( )" on page 189, `PODSDocumentSrc` object's "documentForUrl( )" on page 183

registerDocumentSrc( )

Registers a document source with the document manager.

Interface

`PODSDocumentMgr`

IDL definition

void registerDocumentSrc(PODSDocumentSrc *documentSrc*);

JavaScript synopsis

Not applicable

C synopsis

```
void PODSregisterDocumentSrc(
  PODSDocumentMgr* docmgr,
  PODSDocumentSrc* documentSrc
);
```

Parameters

- ***docmgr*** The `PODSDocumentMgr` object.
- ***documentSrc*** [in] The document source to register.

Returns

None

Remarks

The `registerDocumentSrc( )` and "unregisterDocumentSrc( )" on page 192 methods are used when writing your own document source, via your `PODSPODNew(°)` function. See "Implementing the PODSPodNew( ) function" on page 27.

See also

"unregisterDocumentSrc( )" on page 192

unregisterDocumentSrc( )

Unregister a document source.

Interface

`PODSDocumentMgr`

IDL definition

void unregisterDocumentSrc(PODSDocumentSrc *documentSrc*);

JavaScript synopsis

Not applicable

C synopsis

void **PODSunregisterDocumentSrc(**
  PODSDocumentMgr* *docmgr*,
  PODSDocumentSrc* *documentSrc*
);

Parameters

- ***docmgr*** The `PODSDocumentMgr` object.
- ***documentSrc*** [in] The document source to unregister.

Returns

None

Remarks

The `unregisterDocumentSrc( )` and "registerDocumentSrc( )" on page 191 and methods are used when writing your own document source, via your `PODSPODNew(°)` function. See "Implementing the PODSPodNew( ) function" on page 27.

There is no need to unregister a document source at the end of your application; the `destroy(°)` method, inherited from the `PODSObject` object's "destroy( )" on page 69 automatically performs this function. Only call `unregisterDocumentSrc(°)` to remove a document source from the system before your application ends.

See also

"registerDocumentSrc( )" on page 191

CHAPTER 9

PODS submission-related objects

Contents

Constants for PODS submissions

A form (`PODSSubmission` object) has a submission status, which can have any one of the values represented by the constants listed in the table below. These values are specified in JavaScript as strings, and in PODS as C macros.

Table 1. Submission status constants

Constant	Description
UNSUBMITTED_SUBMIT_STATUS	The form has not been submitted yet.
SUCCESSFUL_SUBMIT_STATUS	The form was submitted successfully.
ERRORED_SUBMIT_STATUS	The form was not submitted successfully.

A `PODSSubmission` object also has a posting method, which can have either of the values represented by the constants listed in "Constants for PODS submissions" on page 196. As with the submission constants, use strings in JavaScript and C macros in PODS.

Table 2. Posting method constants

Constant	Description
SUB_METHOD_POST	Post the contents of the form.
SUB_METHOD_GET	Retrieve the data requested by the form.

The following code example shows how these constants would appear in JavaScript:

```
var subMgr = avantgo.submissionManager;
var submission =
subMgr.createSubmission(
"UNSUBMITTED_SUBMIT_STATUS",
new Date(),
false,
null,
source,
0,
action,
"P", "# " + index++);
```

PODSSubmissionElement object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSSubmission` object's "createSubmissionElement( )" on page 202, or `PODSSubmission` object's "submissionElementForName( )" on page 212
- **Available to:** C, JavaScript

The `PODSSubmissionElement` attributes allow you to get and set the name and value of individual `PODSSubmissionElement` objects. These represent the individual elements on a form.

Summary of PODSSubmissionElement attributes and methods

Description	Attributes and methods
Accessing submission element names and values	"name" on page 197 "value" on page 198

name

The name of the `PODSSubmissionElement` object.

Interface

`PODSSubmissionElement`

IDL definition

attribute PODSString name;

JavaScript synopsis

subelem.**name**

subelem.**name =** *name*

C synopsis

PODSString **PODSgetName(**PODSSubmissionElement* *subelem***);**

void **PODSsetName(**
PODSSubmissionElement* *subelem***,**
PODSString *name*
);

Parameters

- ***subelem*** The `PODSSubmissionElement` object.
- ***name*** [in] Name to set for `PODSSubmissionElement` object.

Returns

Getter: The name of the `PODSSubmissionElement` object.

Setter: None

See also

"value" on page 198

value

The value for the `PODSSubmissionElement` object.

Interface

`PODSSubmissionElement`

IDL definition

attribute PODSString value;

JavaScript synopsis

subelem.**value**

subelem.**value =** *value*

C synopsis

PODSString **PODSgetValue(**PODSSubmissionElement* *subelem*);

void **PODSsetValue(**
PODSSubmissionElement* *subelem*,
PODSString *value*
);

Parameters

- ***subelem*** The `PODSSubmissionElement` object.
- ***value*** [in] Value to set for `PODSSubmissionElement` object.

Returns

Getter: The value for the `PODSSubmissionElement` object.

Setter: None

See also

"name" on page 197

PODSSubmission object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSSubmissionMgr` object's "createSubmission( )" on page 221
- **Available to:** C, JavaScript

`PODSSubmission` represents a form submission request. It also allows creation of its own submission elements.

A `PODSSubmission` object represents a single submission in the M-Business Client's Forms Manager.

Table 3. Summary of PODSSubmission attributes and methods

Description	Attributes and methods
Accessing submission element properties	"actionMethod" on page 200 "actionURL" on page 201 "followOffsiteLinks" on page 205 "formIndex" on page 206 "includeImages" on page 206 "isHidden" on page 207 "linkDepth" on page 208 "maxSize" on page 209 "postData" on page 209 "resultURL" on page 210 "sourceURL" on page 211 "status" on page 212 "submitDate" on page 214 "syncDate" on page 214 "title" on page 215 "trashResponse" on page 216

Description	Attributes and methods
Managing submission elements	“appendSubmissionElement( )” on page 202 “createSubmissionElement( )” on page 202 “deleteSubmissionElement( )” on page 203 “deleteSubmissionElementForIndex( )” on page 204 “submissionElementForName( )” on page 212 “submissionElements” on page 213

actionMethod

The action method for the form (`PODSSubmission` object); the value of the form's (`PODSSubmission` object's) method attribute.

Interface

`PODSSubmission`

IDL definition

attribute PODSString actionMethod;

JavaScript synopsis

sub.**actionMethod**

sub.**actionMethod =** *method*

C synopsis

PODSString **PODSgetActionMethod(**PODSSubmission* *sub*);

void **PODSsetActionMethod(**
PODSSubmission* *sub*,
PODSString *method*
);

Parameters

- ***sub*** The `PODSSubmission` object.
- ***method*** [in] The action method to set: either `SUB_METHOD_POST` or `SUB_METHOD_GET`.

Returns

Getter: The action method for the `PODSSubmission` object: either `SUB_METHOD_POST` or `SUB_METHOD_GET`.

Setter: None

See also

"actionURL" on page 201

actionURL

The action URL for the form (`PODSSubmission` object).

Interface

`PODSSubmission`

IDL definition

attribute PODSString actionURL;

JavaScript synopsis

sub.**actionURL**

sub.**actionURL =** *url*

C synopsis

PODSString **PODSgetActionURL(**PODSSubmission* *sub***);**

void **PODSsetActionURL(**
  PODSSubmission *sub****,**
  PODSString *url*
);

Parameters

- ***sub*** The `PODSSubmission` object.
- ***url*** [in] The value of `actionURL` to set for the `PODSSubmission` object.

Returns

Getter: The action URL for the `PODSSubmission` object.

Setter: None

Remarks

The action URL is the URL specified in the form's (`PODSSubmission` object's) ACTION attribute, and must be fully qualified. This is the URL of the script to perform. This parameter cannot be `NULL`. If it is `NULL`, the form can never be submitted.

See also

"actionMethod" on page 200, "resultURL" on page 210, "sourceURL" on page 211

appendSubmissionElement( )

Appends `PODSSubmissionElement` object to the form (`PODSSubmission` object).

Interface

`PODSSubmission`

IDL definition

```
PODSSubmissionElement appendSubmissionElement(
  PODSSubmissionElement element
);
```

JavaScript synopsis

sub.**appendSubmissionElement(***element***)**

C synopsis

```
PODSSubmissionElement* PODSappendSubmissionElement(
  PODSSubmission* sub,
  PODSSubmissionElement* element
);
```

Parameters

- ***sub*** The `PODSSubmission` object.
- ***element*** [in] The `PODSSubmissionElement` object to append.

Returns

The appended `PODSSubmissionElement` object.

Remarks

The `createSubmissionElement(°)` method does not add the submission element to the submission. You must call `appendSubmissionElement(°)` to add it to the submission, then call `saveSubmission` for your change to take effect.

See also

"createSubmissionElement( )" on page 202, "appendSubmission( )" on page 218, "createSubmission( )" on page 221, "saveSubmission( )" on page 224

createSubmissionElement( )

Creates `PODSSubmissionElement` object.

Interface

`PODSSubmission`

IDL definition

```
PODSSubmissionElement createSubmissionElement(
  PODSString name,
  PODSString value
);
```

JavaScript synopsis

sub.**createSubmissionElement(***name*, *value***)**

C synopsis

```
PODSSubmissionElement* createSubmissionElement(
  PODSSubmission* sub,
  PODSString name,
  PODSString value
);
```

Parameters

- ***sub*** The `PODSSubmission` object.
- ***name*** [in] The name for the `PODSSubmissionElement` object.
- ***value*** [in] The value for the `PODSSubmissionElement` object representing untyped form data to be added to the submission.

Returns

The `PODSSubmissionElement` object created.

Remarks

The `createSubmissionElement()` method does not add the submission element to the submission. You must call `appendSubmissionElement(°)` to add it to the submission, then call `saveSubmission` for your change to take effect.

Note that when creating form data objects, you do not specify the type.

See also

"appendSubmissionElement( )" on page 202, "appendSubmission( )" on page 218, "createSubmission( )" on page 221, "saveSubmission( )" on page 224

deleteSubmissionElement( )

Deletes a `PODSSubmissionElement` object.

Interface

`PODSSubmission`

IDL definition

```
void deleteSubmissionElement(PODSSubmissionElement element);
```


JavaScript synopsis

sub.**deleteSubmissionElement(***element***)**

C synopsis

```
void (PODSdeleteSubmissionElement) (
  PODSSubmission* sub,
  PODSSubmissionElement* element
);
```

Parameters

- ***sub*** The `PODSSubmission` object.
- ***element*** [in] The `PODSSubmissionElement` object to delete.

Returns

Nothing

Remarks

The submission is deleted automatically. There is no need to call `saveSubmission` for your change to take effect.

See also

"deleteSubmissionElementForIndex( )" on page 204, "createSubmissionElement( )" on page 202

deleteSubmissionElementForIndex( )

Deletes a `PODSSubmissionElement` object for a given index value.

Interface

`PODSSubmission`

IDL definition

void deleteSubmissionElementForIndex(PODSInt32 *index*);

JavaScript synopsis

sub.**deleteSubmissionElementForIndex(***index***)**

C synopsis

```
void(PODSdeleteSubmissionElementForIndex(
  PODSSubmission* sub,
  PODSInt32 index
);
```

Parameters

- ***sub*** The `PODSSubmission` object.
- ***index*** The index value for the `PODSSubmissionElement` object to be deleted.

Returns

Nothing

Remarks

The submission is deleted automatically. There is no need to call `saveSubmission` for your change to take effect.

See also

"deleteSubmissionElement( )" on page 203, "createSubmissionElement( )" on page 202

followOffsiteLinks

The setting of `followOffsiteLinks` for the form (`PODSSubmission` object).

Interface

`PODSSubmission`

IDL definition

attribute PODSBoolean followOffsiteLinks;

JavaScript synopsis

sub.**followOffsiteLinks**

sub.**followOffsiteLinks =** *bool*

C synopsis

PODSBoolean **PODSgetFollowOffsiteLinks(**
  PODSSubmission* *sub*
);

void **PODSsetFollowOffsiteLinks(**
  PODSSubmission* *sub*,
  PODSBoolean *bool*
);

Parameters

- ***sub*** The `PODSSubmission` object.
- ***bool*** [in] The value of `followOffsiteLinks` to set for the `PODSSubmission` object: `PODS_TRUE` or `PODS_FALSE` (C Macro); `TRUE` or `FALSE` (JavaScript).

Returns

Getter: `PODS_TRUE` or `TRUE` if `followOffsiteLinks` is set to `PODS_TRUE` or `TRUE`; `PODS_FALSE` or `FALSE` otherwise.

Setter: None

See also

"linkDepth" on page 208

formIndex

The value of `formIndex` for the form (`PODSSubmission` object).

Interface

`PODSSubmission`

IDL definition

attribute PODSInt32 formIndex;

JavaScript synopsis

sub.**formIndex**

sub.**formIndex =** *index*

C synopsis

PODSInt32 **PODSgetFormIndex(**PODSSubmission* *sub***);**

void **PODSsetFormIndex(**
PODSSubmission* *sub***,**
PODSInt32 *index*
);

Parameters

- ***sub*** The `PODSSubmission` object.
- ***index*** [in] The value of `formIndex` to set for the `PODSSubmission` object.

Returns

Getter: The value of `formIndex` for the `PODSSubmission` object.

Setter: None

Remarks

The `formIndex` is the index of the form (`PODSSubmission` object) on the source document page. A page may contain several forms and this is the index of the form in the list of forms on the page (with numbering starting at 0).

includeImages

The value of `includeImages` for the form (`PODSSubmission` object).

Interface

PODSSubmission

IDL definition

attribute PODSBoolean includeImages;

JavaScript synopsis

sub.**includeImages**

sub.**includeImages =** *bool*

C synopsis

PODSBoolean **PODSgetIncludeImages(**PODSSubmission* *sub***);**

void **PODSsetIncludeImages(**
PODSSubmission* *sub***,**
PODSBoolean *bool*
);

Parameters

- ***sub*** The PODSSubmission object.
- ***bool*** [in] The value of includeImages to set for the PODSSubmission object: PODS_TRUE or PODS_FALSE (C Macro); TRUE or FALSE (JavaScript).

Returns

Getter: PODS_TRUE or TRUE if followOffsiteLinks is set to PODS_TRUE or TRUE; PODS_FALSE or FALSE otherwise.

Setter: None

isHidden

The value of isHidden for the form (PODSSubmission object), which determines whether the form appears in the M-Business Client Forms Manager dialog: PODS_TRUE hides the form; PODS_FALSE displays the form.

Interface

PODSSubmission

IDL definition

attribute PODSBoolean isHidden;

JavaScript synopsis

sub.**isHidden**

sub.**isHidden =** *bool*

C synopsis

PODSBoolean **PODSgetIsHidden(**PODSSubmission* *sub***);**

void **PODSsetIsHidden(**
PODSSubmission* *sub***,**
PODSBoolean *bool*
);

Parameters

- ***sub*** The `PODSSubmission` object.
- ***bool*** [in] The value of `isHidden` to set for the `PODSSubmission` object: `PODS_TRUE` or `PODS_FALSE` (C Macro); `TRUE` or `FALSE` (JavaScript).

Returns

Getter: `PODS_TRUE` or `TRUE` if `isHidden` is set to `PODS_TRUE` or `TRUE`; `PODS_FALSE` or `FALSE` otherwise.

Setter: None

linkDepth

The value of `linkDepth` for the form (`PODSSubmission` object).

Interface

`PODSSubmission`

IDL definition

attribute PODSUInt32 linkDepth;

JavaScript synopsis

sub.**linkDepth**

sub.**linkDepth =** *depth*

C synopsis

PODSUInt32 **PODSgetLinkDepth(**PODSSubmission* *sub***);**

void **PODSsetLinkDepth(**
PODSSubmission* *sub***,**
PODSUInt32 *depth*
);

Parameters

- ***sub*** The `PODSSubmission` object.
- ***depth*** [in] The value of `linkDepth` to set.

Returns

Getter: The value of `linkDepth` for the `PODSSubmission` object.

Setter: None

maxSize

The value of `maxSize` for the form (`PODSSubmission` object).

Interface

`PODSSubmission`

IDL definition

attribute PODSUInt32 maxSize;

JavaScript synopsis

sub.**maxSize**

sub.**maxSize =** *size*

C synopsis

PODSUInt32 **PODSDgetMaxSize(**PODSSubmission* *sub***);**

void **PODSsetMaxSize(**
PODSSubmission* *sub***,**
PODSUInt32 *size*
);

Parameters

- ***sub*** The `PODSSubmission` object.

Returns

The value of `maxSize` for the `PODSSubmission` object.

- ***size*** [in] The value of `maxSize` to set.

postData

The value of `postData` for the form (`PODSSubmission` object).

Interface

`PODSSubmission`

IDL definition

attribute PODSString postData;

JavaScript synopsis

sub.**postData**

sub.**postData =** *data*

C synopsis

PODSString **PODSgetPostData(**PODSSubmission* *sub***);**

void **PODSsetPostData(**
PODSSubmission* *sub***,**
PODSString *data*
);

Parameters

- ***sub*** The `PODSSubmission` object.
- ***data*** [in] The value of `postData` to set.

Returns

The value of `postData` for the `PODSSubmission` object.

resultURL

The value of `resultURL` for the form (`PODSSubmission` object).

Interface

`PODSSubmission`

IDL definition

attribute PODSString resultURL;

JavaScript synopsis

sub.**resultURL**

sub.**resultURL =** *url*

C synopsis

PODSString **PODSgetResultURL(**PODSSubmission* *sub***);**

void **PODSsetResultURL(**
PODSSubmission* *sub***,**
PODSString *url*
);

Parameters

- ***sub*** The `PODSSubmission` object.
- ***url*** [in] The value of `resultURL` to set for the `PODSSubmission` object.

Returns

Getter: The value of `resultURL` for the `PODSSubmission` object.

Setter: None

Remarks

The result URL for the form (`PODSSubmission` object) is the absolute URL of the document returned when the form is submitted. This is the page that displays the result of the form's action. If the form has never been submitted, specify `NULL` for this parameter.

See also

"actionURL" on page 201, "sourceURL" on page 211

sourceURL

The value of `sourceURL` for the form (`PODSSubmission` object).

Interface

`PODSSubmission`

IDL definition

attribute PODSString sourceURL;

JavaScript synopsis

sub.**sourceURL**

sub.**sourceURL =** *url*

C synopsis

PODSString **PODSgetSourceURL(**PODSSubmission* *sub***);**

void **PODSsetSourceURL(**
PODSSubmission* *sub***,**
PODSString *url*
);

Parameters

- ***sub*** The `PODSSubmission` object.
- ***url*** [in] The value of `sourceURL` to set.

Returns

Getter: The value of `sourceURL` for the `PODSSubmission` object.

Setter: None

Remarks

The source URL for the form (`PODSSubmission` object) is the absolute URL of the document containing the form. This cannot be `NULL`.

See also

"actionURL" on page 201, "resultURL" on page 210

status

The status for the form (`PODSSubmission` object).

Interface

`PODSSubmission`

IDL definition

attribute PODSString status;

JavaScript synopsis

sub.**status**

sub.**status =** *status*

C synopsis

PODSString **PODSgetStatus(**PODSSubmission* *sub*);

void **PODSsetStatus(**
  PODSSubmission* *sub*,
  PODSString *status*
);

Parameters

- ***sub*** The `PODSSubmission` object.
- ***status*** [in] The status to set. For possible values, see "Constants for PODS submissions" on page 196.

Returns

Getter: The status for the form (`PODSSubmission` object). For possible values, see "Constants for PODS submissions" on page 196.

Setter: None

submissionElementForName( )

Gets the `PODSSubmissionElement` object for a specified name value.

Interface

`PODSSubmission`

IDL definition

PODSSubmissionElement submissionElementForName(PODSString *name*);

JavaScript synopsis

sub.**submissionElementForName(***name***)**

C synopsis

PODSSubmissionElement* **PODSsubmissionElementForName(**
  PODSSubmission* *sub***,**
  PODSString *name*
);

Parameters

- ***sub*** The `PODSSubmission` object.
- ***name*** [in] The name for the `PODSSubmissionElement` object.

Returns

The `PODSSubmissionElement` object for the specified name.

See also

"submissionElements" on page 213

submissionElements

The array of submission element objects for the form (`PODSSubmission` object).

Interface

`PODSSubmission`

IDL definition

readonly PODSArray submissionElements;

JavaScript synopsis

sub.**submissionElements**

C synopsis

PODSArray* **PODSgetSubmissionElements(**PODSSubmission* *sub***);**

Parameters

- ***sub*** The `PODSSubmission` object.

Returns

The array of submission elements for the form.

See also

`PODSSubmissionMgr` object's "submissions( )" on page 225, "submissionElementForName( )" on page 212

submitDate

The submission date for the form (`PODSSubmission` object).

Interface

`PODSSubmission`

IDL definition

attribute PODSDate submitDate;

JavaScript synopsis

sub.**submitDate**

sub.**submitDate =** *date*

C synopsis

PODSDate **PODSgetSubmitDate(**PODSSubmission* *sub***);**

void **PODSsetSubmitDate(**
  PODSSubmission* *sub***,**
  PODSDate *date*
);

Parameters

- ***sub*** The `PODSSubmission` object.
- ***date*** [in] The value of `submitDate` to set.

Returns

Getter: The submission date for the `PODSSubmission` object.

Setter: None

Remarks

Set through the use of a JavaScript Date object.

See also

"syncDate" on page 214

syncDate

The synchronization date for the form (`PODSSubmission` object).

Interface

`PODSSubmission`

IDL definition

attribute PODSDate syncDate;

JavaScript synopsis

sub.**syncDate**

sub.**syncDate =** *date*

C synopsis

PODSDate **PODSgetSyncDate(**PODSSubmission* *sub***);**

void **PODSsetSyncDate(**
PODSSubmission* *sub***,**
PODSDate *date*
);

Parameters

- ***sub*** The `PODSSubmission` object.
- ***date*** [in] The synchronization date to set.

Returns

Getter: The synchronization date for the `PODSSubmission` object.

Setter: None

Remarks

You do not set `syncDate`. The `syncDate` is set after a synchronization has occurred.

See also

"submitDate" on page 214

title

Gets the title for the form (`PODSSubmission` object).

Interface

`PODSSubmission`

IDL definition

attribute PODSString title;

JavaScript synopsis

sub.**title**

sub.**title =** *title*

C synopsis

PODSString **PODSgetTitle(**PODSSubmission* *sub***);**

void **PODSsetTitle(**
PODSSubmission* *sub***,**
PODSString *title*
);

Parameters

- ***sub*** The `PODSSubmission` object.
- ***title*** [in] The title to set. This title may be `NULL`.

Returns

Getter: The title for the `PODSSubmission` object.

Setter: None

Remarks

The title is the form's (`PODSSubmission` object's) title that appears in the Forms Manager dialog in M-Business Client. The title may be `NULL`.

trashResponse

The value of `trashResponse` for the form (`PODSSubmission` object).

Interface

`PODSSubmission`

IDL definition

attribute PODSBoolean trashResponse;

JavaScript synopsis

sub.**trashResponse**

sub.**trashResponse =** *bool*

C synopsis

PODSBoolean **PODSgetTrashResponse(**PODSSubmission* *sub***);**

void **PODSsetTrashResponse(**
PODSSubmission* *sub***,**
PODSBoolean *bool*
);

Parameters

- ***sub*** The `PODSSubmission` object.

- ***bool*** [in] The value of `trashResponse` to set: PODS_TRUE or PODS_FALSE in PODS; TRUE or FALSE in JavaScript.

Returns

Getter: PODS_TRUE or TRUE if `trashResponse` is set to PODS_TRUE or TRUE; PODS_FALSE or FALSE otherwise.

Setter: None

PODSSubmissionMgr object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSAvantGo` object's "submissionMgr" on page 92
- **Available to:** C, JavaScript

The `PODSSubmissionMgr` attributes and methods manage form submissions. You use this object only when creating or deleting a `PODSSubmission` object programmatically.

> **Note**
> If your application is programmatically submitting forms from the same page, you may eventually encounter an out of memory error. To avoid this possibility, first determine the number of submissions at which you encounter the error. Then, in the loop that controls the programmatic submissions, insert code that uses "submissions( )" on page 225 in each iteration to check the number of submissions and reload the application well before the error-tripping number is reached.

The `PODSSubmissionMgr` is the highest level submission-related object with which you interact. It owns all the submissions that are in the system, whether pending or already submitted. These are the same submissions that the user sees on the Forms Manager page, plus any that are hidden.

Table 4. Summary of PODSSubmissionMgr attributes and methods

Description	Attributes and methods
Accessing submissions	"submissions( )" on page 225
Managing submissions	"appendSubmission( )" on page 218 "createMdbcsSubmission( )" on page 219 "deleteSubmissionForIndex( )" on page 224 "createSubmission( )" on page 221 "deleteSubmission( )" on page 223 "deleteSubmissionForIndex( )" on page 224 "saveSubmission( )" on page 224

appendSubmission( )

Appends the `PODSSubmission` object to save it for submission to the server.

Interface

`PODSSubmissionMgr`

IDL definition

PODSSubmission appendSubmission(PODSSubmission *sub*);

JavaScript synopsis

avantgo.*submgr*.**appendSubmission(***sub***)**

C synopsis

PODSSubmission* **PODSappendSubmission(**
PODSSubmissionMgr* *submgr*,
PODSSubmission* *sub*
);

Parameters

- ***submgr*** The `PODSSubmissionMgr` object.
- ***sub*** [in] The `PODSSubmission` object to append.

Returns

Appended `PODSSubmission` object.

Remarks

The `appendSubmission(°)` method does not save the submission. You must call `saveSubmission(°)` to save it so that it is sent to the server.

See also

"createSubmission( )" on page 221, "appendSubmissionElement( )" on page 202, "saveSubmission( )" on page 224

createMdbcsSubmission( )

Creates a `PODSSubmission` object, allowing specification of a character set for the title that is displayed in the M-Business Client Forms Manager.

Interface

`PODSSubmissionMgr`

IDL definition

PODSSubmission createSubmission(
PODSString *subStatus*,
PODSDate *subDate*,
PODSBoolean *isHidden*,
PODSString *resultURL*,
PODSString *sourceURL*,
PODSInt32 *formIndex*,
PODSString *actionURL*,
PODSString *actionMethod*,
PODSString *title*,


```
  PODSUInt16 titleCharset
);
```

JavaScript synopsis

```
avantgo.submgr.createSubmission(
  subStatus,
  subDate,
  isHidden,
  resultURL,
  sourceURL,
  formIndex,
  actionURL,
  actionMethod,
  title,
  titleCharset
)
```

C synopsis

```
PODSSubmission* PODScreateSubmission(
  PODSSubmissionMgr* submgr,
  PODSString subStatus,
  PODSDate subDate,
  PODSBoolean isHidden,
  PODSString resultURL,
  PODSString sourceURL,
  PODSInt32 formIndex,
  PODSString actionURL,
  PODSString actionMethod,
  PODSString title,
  PODSUInt16 titleCharset
);
```

Parameters

- ***submgr*** The `PODSSubmissionMgr` object.
- ***subStatus*** [in] The submission status for the form (`PODSSubmission` object). For possible values, see the Submission status constants table in "Constants for PODS submissions" on page 196.
- ***subDate*** [in] The submission date for the form (`PODSSubmission` object), represented as the number of seconds from the absolute reference date used by the mobile device.
- ***isHidden*** [in] The `isHidden` setting for the form (`PODSSubmission` object), which determines whether the form appears in the M-Business Client Forms Manager dialog:

  - PODS_TRUE hides the form

  - PODS_FALSE displays the form

- ***resultURL*** [in] The result URL for the form (`PODSSubmission` object). This is the absolute URL of the document returned when the form is submitted. This is the page that displays the result of the form's action. If the form has never been submitted, specify NULL for this parameter.
- ***sourceURL*** [in] The source URL for the form (`PODSSubmission` object). This is the absolute URL of the document containing the form. This cannot be NULL.

- ***formIndex*** [in] The index of the form (`PODSSubmission` object) on the source document page. A page may contain several forms and this is the index of the form in the list of forms on the page (with numbering starting at 0).
- ***actionURL*** [in] The URL specified in the form's (`PODSSubmission` object's) ACTION attribute. This is the URL of the script to perform. This parameter cannot be NULL. If it is NULL, the form can never be submitted.
- ***actionMethod*** [in] The value of the form's (`PODSSubmission` object's) method attribute. Possible values are: `SUB_METHOD_POST` or `SUB_METHOD_GET`.
- ***title*** [in] The form's (`PODSSubmission` object's) title. This title appears in the Forms Manager dialog in M-Business Client. You may specify NULL for this parameter.
- ***titleCharset*** [in] The character set to be used in displaying the form's title in the Forms Manager. For valid values, see "Constants to specify a title's character set" on page 60.

Returns

`PODSSubmission` object created.

Remarks

The `createMdbcsSubmission( )` method does not save the submission. You must call `appendSubmission( )` to save it so that it is sent to the server. The source URL and result URL may be accessed when users access the Forms Manager list. If a user selects a submission in the list and clicks **Edit**, M-Business Client displays the source URL to allow the user to edit the submission data. If a user selects an already submitted form from the Forms Manager list and clicks **View**, M-Business Client loads the result page so that the user sees the result of the action.

See also

"createSubmission( )" on page 221, "appendSubmission( )" on page 218, "saveSubmission( )" on page 224

createSubmission( )

Creates a `PODSSubmission` object.

Interface

`PODSSubmissionMgr`

IDL definition

```
PODSSubmission createSubmission(
  PODSString subStatus,
  PODSDate subDate,
  PODSBoolean isHidden,
  PODSString resultURL,
  PODSString sourceURL,
  PODSInt32 formIndex,
  PODSString actionURL,
```



```
  PODSString actionMethod,
  PODSString title
);
```

JavaScript synopsis

```
avantgo.submgr.createSubmission(
  subStatus,
  subDate,
  isHidden,
  resultURL,
  sourceURL,
  formIndex,
  actionURL,
  actionMethod,
  title
)
```

C synopsis

```
PODSSubmission* PODScreateSubmission(
  PODSSubmissionMgr* submgr,
  PODSString subStatus,
  PODSDate subDate,
  PODSBoolean isHidden,
  PODSString resultURL,
  PODSString sourceURL,
  PODSInt32 formIndex,
  PODSString actionURL,
  PODSString actionMethod,
  PODSString title
);
```

Parameters

- ***submgr*** The `PODSSubmissionMgr` object.
- ***subStatus*** [in] The submission status for the form (`PODSSubmission` object). For possible values, see "Constants for PODS submissions" on page 196.
- ***subDate*** [in] The submission date for the form (`PODSSubmission` object), represented as the number of seconds from the absolute reference date used by the mobile device.
- ***isHidden*** [in] The `isHidden` setting for the form (`PODSSubmission` object), which determines whether the form appears in the M-Business Client Forms Manager dialog: `PODS_TRUE` hides the form; `PODS_FALSE` displays the form
- ***resultURL*** [in] The result URL for the form (`PODSSubmission` object). This is the absolute URL of the document returned when the form is submitted. This is the page that displays the result of the form's action. If the form has never been submitted, specify `NULL` for this parameter.
- ***sourceURL*** [in] The source URL for the form (`PODSSubmission` object). This is the absolute URL of the document containing the form. This cannot be `NULL`.

- ***formIndex*** [in] The index of the form (`PODSSubmission` object) on the source document page. A page may contain several forms and this is the index of the form in the list of forms on the page (with numbering starting at 0).
- ***actionURL*** [in] The URL specified in the form's (`PODSSubmission` object's) ACTION attribute. This is the URL of the script to perform. This parameter cannot be `NULL`. If it is `NULL` , the form can never be submitted.
- ***actionMethod*** [in] The value of the form's (`PODSSubmission` object's) method attribute. Possible values are: `SUB_METHOD_POST` or `SUB_METHOD_GET`.
- ***title*** [in] The form's (`PODSSubmission` object's) title. This title appears in the Forms Manager dialog in M-Business Client. You may specify `NULL` for this parameter.

Returns

`PODSSubmission` object created.

Remarks

The `createSubmission(°)` method does not save the submission. You must call `appendSubmission(°)` to save it so that it is sent to the server.

The source URL and result URL may be accessed when users access the Forms Manager list. If a user selects a submission in the list and clicks **Edit**, M-Business Client displays the source URL to allow the user to edit the submission data. If a user selects an already submitted form from the Forms Manager list and clicks **View** , M-Business Client loads the result page so that the user sees the result of the action.

See also

"createMdbcsSubmission( )" on page 219, "appendSubmission( )" on page 218, "saveSubmission( )" on page 224

deleteSubmission( )

Deletes the specified `PODSSubmission` object.

Interface

`PODSSubmissionMgr`

IDL definition

void deleteSubmission(PODSSubmission *sub*);

JavaScript synopsis

avantgo.*submgr*.**deleteSubmission(***sub***)**

C synopsis

void **PODSdeleteSubmission(**
PODSSubmissionMgr* *submgr*,
PODSSubmission* *sub*
);

Parameters

- ***submgr*** The `PODSSubmissionMgr` object.
- ***sub*** [in] The `PODSSubmission` object to be deleted.

Returns

None

See also

"deleteSubmissionForIndex( )" on page 224

deleteSubmissionForIndex( )

Deletes the `PODSSubmission` object for a specified index value.

Interface

`PODSSubmissionMgr`

IDL definition

void deleteSubmissionForIndex(PODSInt32 *index*);

JavaScript synopsis

avantgo.*submgr*.**deleteSubmissionForIndex(***index***)**

C synopsis

void **PODSdeleteSubmissionForIndex(**
  PODSSubmissionMgr* *submgr*,
  PODSInt32 *index*
);

Parameters

- ***submgr*** The `PODSSubmissionMgr` object.
- ***index*** [in] The index value for the `PODSSubmission` object to be deleted. Submissions are numbered in the order that the user (or the POD) entered them.

Returns

None

See also

"deleteSubmission( )" on page 223

saveSubmission( )

Saves the `PODSSubmission` object for a specified index value.

Interface

`PODSSubmissionMgr`

IDL definition

PODSSubmission saveSubmission(PODSSubmission *sub*);

JavaScript synopsis

avantgo.*submgr*.**saveSubmission(***sub***)**

C synopsis

```
PODSSubmission* PODSsaveSubmission(
  PODSSubmissionMgr* submgr,
  PODSSubmission* sub
);
```

Parameters

- ***submgr*** The `PODSSubmissionMgr` object.
- ***sub*** [in] The `PODSSubmission` object to be saved.

Returns

Nothing

Remarks

When you modify a submission or one of its submission elements, you must call `saveSubmission(°)` to cause those changes to be committed.

See also

"createSubmission( )" on page 221, "appendSubmission( )" on page 218

submissions( )

The array of submission objects for the `PODSSubmissionMgr` object.

Interface

`PODSSubmissionMgr`

IDL definition

readonly PODSArray submissions;

JavaScript synopsis

avantgo.*submgr*.**submissions**

C synopsis

```
PODSArray* PODSgetSubmissions(
  PODSSubmissionMgr* submgr
);
```


Parameters

- ***submgr*** The `PODSSubmissionMgr` object.

Returns

The array of submission objects for the `PODSSubmissionMgr` object.

See also

`PODSSubmission` object's "submissionElements" on page 213

CHAPTER 10

PODS browser-related objects

Contents

PODSButton object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSToolbar` object's "createButton( )" on page 234, or any `PODSToolbar` method that gets a button
- **Available to:** C only

The `PODSButton` attributes and methods get and set select properties for buttons on the M-Business Client browser toolbar, as well as animating and setting callback methods for those buttons.

Summary of PODSButton attributes and methods

Description	Attributes and methods
Accessing button's name, visible state, or enabled state	"enabled" on page 229 "visible" on page 230
Setting callback or animating a button	"animate( )" on page 228 "setCallback( )" on page 229

animate( )

Animates the `PODSButton` object.

Interface

`PODSButton`

IDL definition

nometadata void animate( );

JavaScript synopsis

Not applicable

C synopsis

void **PODSanimate(**PODSButton**btn***);**

Parameters

- ***btn*** The `PODSButton` object.

Returns

None

Remarks

`animate( )` works only on Palm OS.

enabled

The state of `enabled` for the `PODSButton` object.

Interface

`PODSButton`

IDL definition

attribute PODSBoolean enabled;

JavaScript synopsis

btn.**enabled**

btn.**enabled =** *bool*

C synopsis

PODSBoolean **PODSgetEnabled(**PODSButton* *btn***);**

void **PODSsetEnabled(**
PODSButton* *btn***,**
PODSBoolean *bool*
);

Parameters

- ***btn*** The `PODSButton` object.
- ***bool*** [in] The state to set: `PODS_TRUE` or `PODS_FALSE`.

Returns

Getter: `PODS_TRUE` if `enabled` is set to `PODS_TRUE`; `PODS_FALSE` otherwise.

Setter: None

Remarks

If `enabled` is set to `PODS_TRUE` the button is enabled. Otherwise the button is disabled.

setCallback( )

Sets the callback method for the `PODSButton` object.

Interface

`PODSButton`

IDL definition

```
nometadata void setCallback(
  PODSMethod method,
  PODSObject targetobj,
  any closure
);
```

JavaScript synopsis

Not applicable

C synopsis

```
void PODSsetCallback(
  PODSButton* btn,
  PODSMethod method,
  PODSObject* targetobj,
  void* closure
);
```

Parameters

- ***btn*** The `PODSButton` object.
- ***method*** [in] The callback method to be invoked.
- ***targetobj*** [in] The object on which `method` will be invoked.
- ***closure*** [out] Data to be passed back when user clicks the `PODSButton` object.

Returns

None

visible

The state of `visible` for the `PODSButton` object.

Interface

`PODSButton`

IDL definition

```
attribute PODSBoolean visible;
```

JavaScript synopsis

btn.**visible**

btn.**visible =** *bool*

C synopsis

PODSBoolean **PODSgetVisible(**PODSButton* *btn*);


```
void setVisible(
  PODSButton* btn,
  PODSBoolean bool
);
```

Parameters

- ***btn*** The `PODSButton` object.
- ***bool*** [in] The state to set: `PODS_TRUE` or `PODS_FALSE`.

Returns

Getter: `PODS_TRUE` if `visible` is set to `PODS_TRUE`; `PODS_FALSE` otherwise.

Setter: None

Remarks

If `visible` is set to `PODS_TRUE`, the button is visible on the toolbar. Otherwise it is not. Whether a gap appears in the toolbar in the position of the invisible button depends on the platform.

PODSToolbar object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSWindow` object's "toolbar" on page 250
- **Available to:** C, JavaScript

The `PODSToolbar` attributes and methods customize the M-Business Client toolbar and manipulate its buttons.

Summary of PODSToolbar attributes and methods

Description	Attributes and methods
Structuring the toolbar	"buttonCount" on page 232 "createButton( )" on page 234 "deleteButton( )" on page 235 "title" on page 235
Getting button objects	"buttonForIndex( )" on page 233 "buttonForName( )" on page 233

buttonCount

The number of buttons in the `PODSToolbar` object on the M-Business Client toolbar.

Interface

`PODSToolbar`

IDL definition

readonly attribute PODSUInt16 buttonCount;

JavaScript synopsis

tbar.**buttonCount**

C synopsis

PODSUInt16 **PODSbuttonCount(**PODSToolbar* *tbar*);

Parameters

- ***tbar*** The `PODSToolbar` object.

Returns

The number of buttons in the `PODSToolbar` object.

See also

"buttonForIndex( )" on page 233, "buttonForName( )" on page 233

buttonForIndex( )

Gets the `PODSButton` object from the M-Business Client toolbar for a specified index value.

Interface

`PODSToolbar`

IDL definition

PODSButton buttonForIndex(PODSUInt16 *index*);

JavaScript synopsis

tbar.**buttonForIndex(***index***)**

C synopsis

```
PODSButton* PODSbuttonForIndex(
  PODSToolbar* tbar,
  PODSUInt16 index
);
```

Parameters

- ***tbar*** The `PODSToolbar` object.
- ***index*** [in] The index value for the desired `PODSButton` object.

Returns

The `PODSButton` object for the specified index value.

See also

"buttonForName( )" on page 233, "buttonCount" on page 232

buttonForName( )

Gets the `PODSButton` object from the M-Business Client toolbar for a specified name.

Interface

`PODSToolbar`

IDL definition

PODSButton buttonForName(PODSString *name*);

JavaScript synopsis

tbar.**buttonForName(***name***)**

C synopsis

PODSButton* **PODSbuttonForName(**
  PODSToolbar* *tbar***,**
  PODSString *name*
);

Parameters

- ***tbar*** The `PODSToolbar` object.
- ***name*** [in] The name of the desired `PODSButton` object.

Returns

The `PODSButton` object for the specified name.

See also

"buttonForIndex( )" on page 233, "buttonCount" on page 232

createButton( )

Makes a new `PODSButton` object on the M-Business Client toolbar.

Interface

`PODSToolbar`

IDL definition

PODSButton createButton( );

JavaScript synopsis

tbar.**createButton( )**

C synopsis

PODSButton* **PODScreateButton(**PODSToolbar* *tbar***);**

Parameters

- ***tbar*** The `PODSToolbar` object.

Returns

The new `PODSButton` object.

See also

"deleteButton( )" on page 235

deleteButton( )

Removes the specified `PODSButton` object from the M-Business Client toolbar.

Interface

`PODSToolbar`

IDL definition

void deleteButton(PODSButton *button*);

JavaScript synopsis

tbar.**removeButton(***button***)**

C synopsis

void **PODSdeleteButton(**
PODSToolbar* *tbar***,**
PODSButton* *button*
);

Parameters

- ***tbar*** The `PODSToolbar` object.
- ***button*** [in] The `PODSButton` object to delete.

Returns

None

See also

"createButton( )" on page 234

title

The title for the `PODSToolbar` object on the M-Business Client toolbar.

Interface

`PODSToolbar`

IDL definition

attribute PODSString title;

JavaScript synopsis

tbar.**title**

tbar.**title =** *name*

C synopsis

PODSString **PODSgetTitle(**PODSToolbar* *tbar***);**


```
void PODSsetTitle(
  PODSToolbar* tbar,
  PODSString name
);
```

Parameters

- ***tbar*** The `PODSToolbar` object.
- ***name*** [in] The title to set.

Returns

Getter: The title for the `PODSToolbar` object.

Setter: None

PODSWindow object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSAvantGo` object's "window" on page 93
- **Available to:** C, JavaScript

The attributes and methods in the `PODSWindow` interface manage a PODS application's M-Business Client browser window. These methods access the window's read-only attributes, get and set the URL displayed in the window, display a variety of window dialogs, and navigate within the window's browse history.

`PODSWindow` is equivalent to the JavaScript `Window` object.

Summary of PODSWindow attributes and methods

Description	Attributes and methods
Accessing window's read-only attributes	"currentSubmissionForForm" on page 240 "document" on page 242 "event" on page 242 "history" on page 245 "menu" on page 246 "navigator" on page 246 "screen" on page 248 "self" on page 248 "toolbar" on page 250 "top" on page 251 "window" on page 251
Displaying window dialogs	"alert( )" on page 238 "confirm( )" on page 240 "dispatchEvent" on page 241 "prompt( )" on page 247

Description	Attributes and methods
Navigating window contents	“back( )” on page 239 “forward( )” on page 243 “home( )” on page 245 “getLocation( )” on page 244 “setLocation( )” on page 249
Miscellaneous	“avantgo” on page 239 “fullScreen” on page 243 “showBusy” on page 249

alert( )

Displays an alert box with a message and an OK button in the M-Business Client browser.

Interface

`PODSWindow`

IDL definition

void alert(PODSString *msg*);

JavaScript synopsis

window.**alert(***msg***)**

C synopsis

void **PODSalert(**
  PODSWindow* *window*,
  PODSString *msg*
);

Parameters

- ***window*** The `PODSWindow` object.
- ***msg*** [in] The message to display in the dialog.

Returns

None

Remarks

Use to display a message that does not require a user decision.

See also

“confirm( )” on page 240, “prompt( )” on page 247

avantgo

The PODSAvantGo object for the window.

Interface

PODSWindow

IDL definition

readonly attribute PODSAvantGo avantgo;

JavaScript synopsis

window.**avantgo( )**

C synopsis

PODSAvantGo* **PODSgetAvantgo(**PODSWindow* *window*);

Parameters

- ***window*** The PODSWindow object.

Returns

PODSAvantgo object for the window.

back( )

Causes the M-Business Anywhere client browser to go back one page in the user's browse history.

Interface

PODSWindow

IDL definition

void back( );

JavaScript synopsis

window.**back( )**

C synopsis

void **PODSback(**PODSWindow* *window*);

Parameters

- ***window*** The PODSWindow object.

Returns

None

Remarks

Equivalent to the user pressing the browser’s Back button.

See also

"forward( )" on page 243, "home( )" on page 245, "getLocation( )" on page 244, "setLocation( )" on page 249

confirm( )

Displays a Confirm dialog with the specified message and OK and Cancel buttons in the M-Business Client browser.

Interface

`PODSWindow`

IDL definition

PODSBoolean confirm(PODSString *question*);

JavaScript synopsis

window.**confirm(***question***)**

C synopsis

PODSBoolean **PODSconfirm(**
  PODSWindow* *window*,
  PODSString *question*
);

Parameters

- ***window*** The `PODSWindow` object.
- ***question*** [in] The message to display in the dialog.

Returns

`PODS_TRUE`

If the user clicked the **OK** button.

`PODS_FALSE`

Otherwise. (The user clicked the **Cancel** button.)

See also

"alert( )" on page 238, "prompt( )" on page 247

currentSubmissionForForm

Returns the corresponding submission for a form that is being edited.

Interface

`PODSWindow`

IDL definition

PODSSubmission currentSubmissionForForm(ADOMHTMLFormElement *form*);

JavaScript synopsis

window.**currentSubmissionForForm**

C synopsis

PODSSubmission* **PODScurrentSubmissionForForm(**
 PODSWindow* *window***,**
 ADOMHTMLFormElement* *form*
);

Parameters

- ***window*** The `PODSWindow` object.
- ***form*** The specified form element object.

Returns

The current submission object for the specified form element.

dispatchEvent

Sends an event to the browser.

Interface

`PODSWindow`

IDL definition

PODSBoolean dispatchEvent (ADOMEvent event, PODSString type);

JavaScript synopsis

window.**dispatchEvent (***event***,** *type***)**

C synopsis

PODSBoolean* **PODSdispatchEvent(**
 PODSWindow* *window***,**
 ADOMEvent* *event***,**
 PODSString *type*
);

Parameters

- ***window*** The `PODSWindow` object.
- ***event*** The event to be sent to the browser.
- ***type*** The name of the event that the event handler will use in determining whether to handle the event.

Returns

PODS_TRUE if event has been handled; PODS_FALSE otherwise.

Remarks

You can add an event handler to any node in the DOM that will respond to a specific event name.

document

Returns a read-only reference to the document object contained in the window.

Interface

PODSWindow

IDL definition

read-only attribute ADOMHTMLDocument document;

JavaScript synopsis

window.**document**

C synopsis

ADOMHTMLDocument **PODSgetDocument(**PODSWindow* *window***);**

Parameters

- ***window*** The PODSWindow object.

Returns

Read-only reference to window's document object.

event

Returns a read-only pointer to the current event being processed by the document.

Interface

PODSWindow

IDL definition

read-only attribute ADOMEvent event;

JavaScript synopsis

window.**event**

window.**event =** *event*

C synopsis

ADOMEvent* **PODSgetEvent(**PODSWindow* *window***);**

Parameters

- ***window*** The `PODSWindow` object.

Returns

Read-only pointer to the current event being processed by the document.

forward( )

Causes the M-Business Anywhere client browser to go forward one page in the user's browse history.

Interface

`PODSWindow`

IDL definition

void forward( );

JavaScript synopsis

window.**forward( )**

C synopsis

void **PODSforward(**PODSWindow* *window***)**

Parameters

- ***window*** The `PODSWindow` object.

Returns

None

Remarks

Equivalent to the user pressing the browser's Forward button.

See also

"back( )" on page 239, "home( )" on page 245, "getLocation( )" on page 244, "setLocation( )" on page 249

fullScreen

The state of `fullScreen` for the `PODSWindow` object.

Interface

`PODSWindow`

IDL definition

attribute PODSBoolean fullScreen;

JavaScript synopsis

window.**fullScreen**

window.**fullScreen =** *bool*

C synopsis

PODSBoolean **getFullScreen(**PODSWindow* *window***);**

void **setFullScreen(**
PODSWindow* *window***,**
PODSBoolean *bool*
);

Parameters

- ***window*** The `PODSWindow` object.
- ***bool*** [in] The state to set: `PODS_TRUE` or `PODS_FALSE`.

Returns

Getter : `PODS_TRUE` if enabled is set to `PODS_TRUE`; `PODS_FALSE` other-wise.

Setter : None

Remarks

Equivalent to the user setting the browser's Full Screen option.

getLocation( )

Gets the `PODSLocation` object for the current M-Business Client browser window.

Interface

`PODSWindow`

IDL definition

PODSLocation getLocation( );

JavaScript synopsis

window.**getLocation( )**

C synopsis

PODSLocation* **PODSgetLocation(**PODSWindow* *window***);**

Parameters

- ***window*** The `PODSWindow` object.

Returns

The `PODSLocation` object for the current M-Business Client browser window.

See also

"setLocation( )" on page 249, "back( )" on page 239, "forward( )" on page 243, "home( )" on page 245

history

Gets a `PODSHistory` object containing the user's browse history for the window.

Interface

`PODSWindow`

IDL definition

read-only attribute PODSHistory history;

JavaScript synopsis

window.**history**

C synopsis

PODSHistory* **PODSgetHistory(**PODSWindow* *window***);**

Parameters

- ***window*** The `PODSWindow` object.

Returns

`PODSHistory` object for the M-Business Client browser window.

home( )

Causes the M-Business Client browser to go to the home page. This is the page that lists all of the channels the user has subscribed to.

Interface

`PODSWindow`

IDL definition

void home( );

JavaScript synopsis

window.**home( )**

C synopsis

void **PODShome(**PODSWindow* *window***);**

Parameters

- ***window*** The `PODSWindow` object.

Returns

None

Remarks

Equivalent to the user pressing the browser's Home button.

See also

"back( )" on page 239, "forward( )" on page 243, "home( )" on page 245, "getLocation( )" on page 244

menu

Gets the `PODSMenu` object for the M-Business Client browser window.

Interface

`PODSWindow`

IDL definition

readonly attribute PODSMenu menu;

JavaScript synopsis

window.**menu**

C synopsis

PODSMenu* **PODSgetMenu(**PODSWindow* *window*);**

Parameters

- ***window*** The `PODSWindow` object.

Returns

`PODSMenu` object for the M-Business Client browser window.

navigator

Gets the `PODSNavigator` object (M-Business Client application) for the M-Business Client browser window.

Interface

`PODSWindow`

IDL definition

readonly attribute PODSNavigator navigator;

JavaScript synopsis

window.**navigator**

C synopsis

PODSNavigator* **PODSgetNavigator(**PODSWindow* *window***);**

Parameters

- ***window*** The `PODSWindow` object.

Returns

`PODSNavigator` object for the M-Business Client browser window.

prompt( )

Displays a Prompt dialog with the specified message and an input field.

Interface

`PODSWindow`

IDL definition

```
PODSString prompt(
  PODSString msg,
  PODSString def
);
```

JavaScript synopsis

window.**prompt**(*msg*, *def***)**

C synopsis

```
PODSString PODSprompt(
  PODSWindow* window,
  PODSString msg,
  PODSString def
);
```

Parameters

- ***window*** The `PODSWindow` object.
- ***msg*** [in] The message to display in the dialog.
- ***def*** [in] Default value to place in the text field.

Returns

Data that the user entered in response to the prompt; `NULL` if the user clicked the **Cancel** button.

See also

"alert( )" on page 238, "confirm( )" on page 240

screen

Gets the PODSScreen object for the M-Business Client browser window.

Interface

PODSWindow

IDL definition

read-only attribute PODSScreen screen;

JavaScript synopsis

window.**screen**

C synopsis

PODSScreen* **PODSgetScreen(**PODSWindow* *window***);**

Parameters

- ***window*** The PODSWindow object.

Returns

The PODSScreen object for the M-Business Client browser window.

self

A synonym for "window" on page 251.

Interface

PODSWindow

IDL definition

read-only attribute PODSWindow self;

JavaScript synopsis

window.**self**

C synopsis

PODSWindow* **PODSgetSelf(**PODSWindow* *window***);**

Parameters

- ***window*** The PODSWindow object.

Returns

The PODSWindow object.

Remarks

`getSelf( )` and `getWindow( )` perform the same function. The two are provided to support JavaScript conventions. Use `getSelf( )` when you want to distinguish a window property from a form or form element of the same name, to make your code more readable.

See also

"window" on page 251

setLocation( )

Loads and displays the contents of the specified URL in the M-Business Client browser window.

Interface

`PODSWindow`

IDL definition

void setLocation(PODSString *url*);

JavaScript synopsis

window.**setLocation(***url***)**

C synopsis

void **PODSsetLocation(**
  PODSWindow* *window*,
  PODSString *url*
);

Parameters

- ***window*** The `PODSWindow` object.
- ***url*** [in] The URL for the page to be displayed.

Returns

None

See also

"getLocation( )" on page 244, "back( )" on page 239, "forward( )" on page 243, "home( )" on page 245

showBusy

Makes a busy indicator appear when set to true.

Interface

`PODSWindow`

IDL definition

attribute PODSBoolean showBusy;

JavaScript synopsis

window.**showBusy**

window.**showBusy =** *bool***;**

C synopsis

PODSBoolean **PODSgetShowBusy(**PODSWindow* *window***)**

void **PODSsetShowBusy(**
PODSWindow* *window***,**
PODSBoolean *bool*
)

Parameters

- ***window*** The `PODSWindow` object.
- ***bool*** [in] The state to set: `PODS_TRUE` or `PODS_FALSE`.

Returns

Getter: `PODS_TRUE` if `showBusy` is set to `PODS_TRUE`; `PODS_FALSE` otherwise.

Setter: None

Remarks

The `showBusy` attribute is automatically set to `PODS_FALSE` at the end of any eventhandler (`onclick`, `onload`, etc.).

toolbar

Gets the `PODSToolbar` object for the M-Business Client browser window.

Interface

`PODSWindow`

IDL definition

read-only attribute PODSToolbar toolbar;

JavaScript synopsis

window.**toolbar**

C synopsis

PODSToolbar* **PODSgetToolbar(**PODSWindow* *window***);**

Parameters

- ***window*** The `PODSWindow` object.

Returns

The `PODSToolbar` object for the M-Business Client browser window.

top

Gets the topmost browser window in M-Business Client.

Interface

`PODSWindow`

IDL definition

read-only attribute PODSWindow top;

JavaScript synopsis

window.**top**

C synopsis

PODSWindow* **PODSgetTop(**PODSWindow* *window***);**

Parameters

- ***window*** The `PODSWindow` object.

Returns

The topmost browser window in M-Business Client.

window

Gets the current window object from the M-Business Client browser.

Interface

`PODSWindow`

IDL definition

readonly attribute PODSWindow window;

JavaScript synopsis

window.**window**

C synopsis

PODSWindow* **PODSgetWindow(**PODSWindow* *window***);**

Parameters

- ***window*** The `PODSWindow` object.

Returns

`PODSWindow` object for the current window.

Remarks

`getWindow( )` and `getSelf( )` perform the same function. The two are provided to support JavaScript conventions. Use to distinguish a property of the window object from a form or form element of the same name, to make your code more readable.

See also

"self" on page 248

PODSHistory object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSWindow` object's "history" on page 245
- **Available to:** C, JavaScript

The `PODSHistory` attributes and methods return information about the M-Business Client user's browse history and cause the browser to navigate to specified points within that browse history.

`PODSHistory` is equivalent to the JavaScript `History` object.

Summary of PODSHistory attributes and methods

Description	Attributes and methods
Getting information about browse history	"current" on page 254 "length" on page 256 "next" on page 256 "previous" on page 257
Navigating within browse history	"back( )" on page 253 "forward( )" on page 254 "go( )" on page 255 "item( )" on page 255

back( )

Causes the M-Business Client browser to go back one page in the user's browse history.

Interface

`PODSHistory`

IDL definition

void back( );

JavaScript synopsis

hist.**back( )**

C synopsis

void **PODSback(**PODSHistory* *hist***);**

Parameters

- ***hist*** The `PODSHistory` object.

Returns

None

current

The complete URL for the currently displayed page.

Interface

`PODSHistory`

IDL definition

read-only attribute PODSString current;

JavaScript synopsis

hist.**current**

C synopsis

PODSString **PODSgetCurrent(**PODSHistory* *hist***);**

Parameters

- ***hist*** The `PODSHistory` object.

Returns

Complete URL for the currently displayed page.

forward( )

Causes the M-Business Client browser to go forward one page in the user's browse history.

Interface

`PODSHistory`

IDL definition

void forward( );

JavaScript synopsis

hist.**forward( )**

C synopsis

void **PODSforward(**PODSHistory* *hist***);**

Parameters

- ***hist*** The `PODSHistory` object.

Returns

None

go( )

Causes the M-Business Client browser to go back or forward to a relative position in the user's browse history.

Interface

`PODSHistory`

IDL definition

void go(PODSInt32 *relativePosition*);

JavaScript synopsis

hist.**go(***relativePosition***)**

C synopsis

void **PODSgo(**
PODSHistory* *hist*,
PODSInt32 *relativePosition*
);

Parameters

- ***hist*** The `PODSHistory` object.
- ***relativePosition*** [in] The position in the browse history, relative to the current position, to be displayed. Negative numbers move backward; e.g., -3 would move back three positions. Positive numbers move forward; e.g., 3 would move forward three positions.

Returns

None

item( )

Returns the URL for the page at a specified position in the user's browse history.

Interface

`PODSHistory`

IDL definition

PODSString item(PODSInt32 *position*);

JavaScript synopsis

hist.**item(***position***)**

C synopsis

PODSString **PODSitem(**
  PODSHistory* *hist***,**
  PODSInt32 *position*
);

Parameters

- ***hist*** The `PODSHistory` object.
- ***position*** [in]

Returns

URL for the page at the specified position.

length

The total number of pages in the user's browse history.

Interface

`PODSHistory`

IDL definition

read-only attribute PODSUInt32 length;

JavaScript synopsis

hist.**length**

C synopsis

PODSUInt32 **PODSgetLength(**PODSHistory* *hist***);**

Parameters

- ***hist*** The `PODSHistory` object.

Returns

Total number of pages in the user's browse history.

next

The complete URL for the next page in the user's browse history.

Interface

`PODSHistory`

IDL definition

read-only attribute PODSString next;

JavaScript synopsis

hist.**next**

C synopsis

PODSString **PODSgetNext(**PODSHistory* *hist***);**

Parameters

- ***hist*** The `PODSHistory` object.

Returns

Complete URL for the next page in the user's browse history.

previous

Returns the complete URL for the previous page in the user's browse history.

Interface

`PODSHistory`

IDL definition

read-only attribute PODSString previous;

JavaScript synopsis

hist.**previous**

C synopsis

PODSString **PODSgetPrevious(**PODSHistory* *hist***);**

Parameters

- ***hist*** The `PODSHistory` object.

Returns

Complete URL for the previous page in the user's browse history.

PODSLocation object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSWindow` object's "getLocation( )" on page 244
- **Available to:** C, JavaScript

The `PODSLocation` attributes and methods manage parameters related to the host for the page currently being displayed in the M-Business Client browser.

`PODSLocation` is equivalent to the JavaScript `Location` object.

Summary of PODSLocation attributes and methods

Description	Attributes and methods
Accessing location properties	"hash" on page 258 "host" on page 259 "hostname" on page 260 "href" on page 261 "pathname" on page 262 "port" on page 262 "protocol" on page 263 "search" on page 264
Reloading/replacing location	"reload( )" on page 265 "replace( )" on page 265

hash

The anchor name portion (string beginning with hash mark, #) of the current URL for the `PODSLocation` object.

Interface

`PODSLocation`

IDL definition

attribute PODSString hash;

JavaScript synopsis

loc.**hash**

loc.**hash =** *hash*

C synopsis

PODSString **PODSgetHash(**PODSLocation* *loc***);**

void **PODSsetHash(**
PODSLocation* *loc***,**
PODSString *hash*
);

Parameters

- ***loc*** The `PODSLocation` object.
- ***hash*** [in] The hash to set.

Returns

Getter: The anchor name portion (string beginning with hash mark, #) of the current URL.

Setter: None

host

The host (string specifying the server name, subdomain, and domain name) for the `PODSLocation` object.

Interface

`PODSLocation`

IDL definition

attribute PODSString host;

JavaScript synopsis

loc.**host**

loc.**host =** *host*

C synopsis

PODSString **PODSgetHost(**PODSLocation* *loc***);**

void **PODSsetHost(**
PODSLocation* *loc***,**
PODSString *host*
);

Parameters

- ***loc*** The `PODSLocation` object.
- ***host*** [in] The value for `host` to set.

Returns

The host for the `PODSLocation` object,

Remarks

String is returned as it appears in the browser's location box; may be relative or absolute, IP and/or DNS address.

hostname

The host name for the `PODSLocation` object.

Interface

`PODSLocation`

IDL definition

attribute PODSString hostname;

JavaScript synopsis

loc.**hostname**

loc.**hostname =** *hostname*

C synopsis

PODSString **PODSgetHostname(**PODSLocation* *loc***);**

void **PODSsetHostname(**
PODSLocation* *loc***,**
PODSString *hostname*
);

Parameters

- ***loc*** The `PODSLocation` object.
- ***hostname*** [in] The host name to set. This is the minimal address required to reach the host over the Internet.

Returns

Getter: The host name for the `PODSLocation` object.

Setter: None

Remarks

Host name includes server name, subdomain, domain, and port number. This is the minimal address required to reach the host over the Internet.

See also

"hash" on page 258, "host" on page 259, "href" on page 261, "pathname" on page 262, "port" on page 262, "protocol" on page 263, "search" on page 264

href

Gets the value for `href` for the `PODSLocation` object.

Interface

`PODSLocation`

IDL definition

attribute PODSString href;

JavaScript synopsis

loc.**href**

loc.**href =** *href*

C synopsis

PODSString **PODSgetHref(**PODSLocation* *loc***);**

void **PODSsetHref(**
  PODSLocation* *loc***,**
  PODSString *href*
);

Parameters

- ***loc*** The `PODSLocation` object.
- ***href*** [in] The value for `href` to set. This is a fully qualified Internet address.

Returns

The value for `href` for the `PODSLocation` object.

Remarks

This is a fully qualified Internet address, the complete URL of the page being displayed. Other `PODSLocation` object properties are substrings of the `href` property. Changing the `href` property for a window correctly updates all of the other location properties.

See also

"href" on page 261, "hash" on page 258, "host" on page 259, "pathname" on page 262, "port" on page 262, "protocol" on page 263, "search" on page 264

pathname

The value of `pathname` for the `PODSLocation` object.

Interface

`PODSLocation`

IDL definition

attribute PODSString pathname;

JavaScript synopsis

loc.**pathname**

loc.**pathname =** *pathname*

C synopsis

PODSString **PODSgetPathname(**PODSLocation* *loc***);**

void **PODSsetPathname(**
PODSLocation* *loc***,**
PODSString *pathname*
);

Parameters

- ***loc*** The `PODSLocation` object.
- ***pathname*** [in] The value for `pathname` for the `PODSLocation` object.

Returns

Getter: The value of `pathname` for the `PODSLocation` object.

Setter: None

Remarks

`pathname` specifies the URL-path portion of the URL, omitting any `hash` (anchor) portion present.

See also

"href" on page 261, "hash" on page 258

port

The port for the `PODSLocation` object.

Interface

`PODSLocation`

IDL definition

attribute PODSString port;

JavaScript synopsis

loc.**port**

loc.**port =** *port*

C synopsis

PODSString **PODSgetPort(**PODSLocation* *loc***);**

void **PODSsetPort(**
PODSLocation* *loc***,**
PODSString *port*
);

Parameters

- ***loc*** The `PODSLocation` object.
- ***port*** [in] The port to set.

Returns

Getter: The port for the `PODSLocation` object.

Setter: None

Remarks

The port is the communications port that the server uses.

protocol

The protocol for the `PODSLocation` object.

Interface

`PODSLocation`

IDL definition

attribute PODSString protocol;

JavaScript synopsis

loc.**protocol**

loc.**protocol =** *protocol*

C synopsis

PODSString **PODSgetProtocol(**PODSLocation**loc***);**

void **PODSsetProtocol(**
PODSLocation* *loc***,**
PODSString *protocol*
);

Parameters

- ***loc*** The `PODSLocation` object.
- ***protocol*** [in] The protocol to set.

Returns

Getter: The protocol for the `PODSLocation` object.

Setter: None

Remarks

The protocol is the beginning of the URL, up to and including the first colon. For example, you can use the `HTTP:` or `HTTPS:` protocols.

See also

"href" on page 261

search

The value of the search portion of the URL for the `PODSLocation` object.

Interface

`PODSLocation`

IDL definition

attribute PODSString search;

JavaScript synopsis

loc.**search**

loc.**search =** *search*

C synopsis

PODSString **PODSgetSearch(**PODSLocation* *loc***);**

```
void PODSsetSearch(
  PODSLocation* loc,
  PODSString search
);
```

Parameters

- ***loc*** The `PODSLocation` object.
- ***search*** [in] The value of `search` to set.

Returns

Getter: The value of `search` for the `PODSLocation` object.

Setter: None

Remarks

The `search` attribute is a portion of the URL, beginning with a question mark, that specifies any query information.

See also

"href" on page 261

reload( )

Reloads the `PODSLocation` object (the window's current document).

Interface

`PODSLocation`

IDL definition

void reload(PODSBoolean *force*);

JavaScript synopsis

loc.**reload(***force***)**

C synopsis

void **PODSreload(**
  PODSLocation* *loc*,
  PODSBoolean *force*
);

Parameters

- ***loc*** The `PODSLocation` object.
- ***force*** [in] If set to `PODS_TRUE`, the reload is forced. Otherwise it is not.

Returns

None

Remarks

The window's current document is the document specified by `href`. `reload(°)` does not change the user's browse history.

See also

"replace( )" on page 265

replace( )

Replaces the `PODSLocation` object with a `PODSLocation` object for a different URL.

Interface

`PODSLocation`

IDL definition

void replace(PODSString *url*);

JavaScript synopsis

loc.**replace(***url***)**

C synopsis

void **PODSreplace(**
  PODSLocation**loc*,**
  PODSString *url*
);

Parameters

- ***loc*** The `PODSLocation` object.
- ***url*** [in] URL for the replacement `PODSLocation` object.

Returns

None

Remarks

The `replace( )` method also replaces the current history entry with the new URL; the browse history is otherwise unchanged.

See also

"reload( )" on page 265

PODSMenu object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSWindow` object's "menu" on page 246
- **Available to:** C, JavaScript

Each of the `PODSMenu` attributes returns the object for a different M-Business Client menu item.

JavaScript Developers do not need to instantiate the `PODSMenu` object before using it.

To use the PODSMenu object from C

1. Include *PODSWindow.h*, *PODSAvantgo.h*, and *PODSMenu.h* header files.
2. Store the `PODSAvantGo` object pass-in from the client application to your `PODSPodNew()` or your `objectForName()` implementation.
3. Acquire a pointer to `PODSWindow` by calling `PODSgetWindow(PODSAvantgoObj)`
4. Acquire a pointer to `PODSMenu` by calling `PODSgetMenu(PODSWindowObj)`.

Note
Some menu items are not available across all M-Business Client platforms. For example, the Home menu item is not available on Windows XP devices. An attempt to access a missing menu item simply results in a no-op and does not cause any exceptions. Use the "isAvail( )" on page 285 method can use to test for the availability of a given menu item.

Note
The valid labels for *menu_item* in the table above are based on their respective labels on the in M-Business Client on the Windows Mobile 5 and higher platform family. Some menu items have different labels on other M-Business Client platforms. For example, the Page Options menu item is called Preferences, or just Options, on some platforms.

Summary of PODSMenu attributes and methods

Description	Attributes and methods
Access and set main menu attributes	"isAvail()" on page 277 "label" on page 278 "remove( )" on page 280

Description	Attributes and methods
Access menu item objects	“about” on page 269 “addBookmark” on page 269 “back” on page 270 “bookmarkManager” on page 270 “cacheManager” on page 271 “channelManager” on page 271 “copy” on page 272 “cut” on page 272 “exit” on page 273 “find” on page 273 “findNext” on page 274 “findPrevious” on page 274 “formManager” on page 275 “forward” on page 275 “fullScreen” on page 276 “goHome” on page 276 “help” on page 277 “openPage” on page 278 “pageOption” on page 279 “paste” on page 279 “reloadPage” on page 280 “selectAll” on page 281 “serverOption” on page 281 “syncAll” on page 282 “workOffline” on page 282 “workOnline” on page 283

***Note**
These functions are only applicable to platforms where the user must display the M-Business Client menus. For example, on Windows Mobile 5, the user must press a hardware button to display the menus. On most platforms, the menus are always displayed.

about

Accesses the object for the About item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem about;

JavaScript synopsis

menu.**about**

C synopsis

PODSMenuItem **PODSgetAbout(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the About item in the M-Business Client menus.

addBookmark

Accesses the object for the Add Bookmark item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem addBookmark;

JavaScript synopsis

menu.**addBookmark**

C synopsis

PODSMenuItem* **PODSgetAddBookmark(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Add Bookmark item in the M-Business Client menus.

back

Accesses the object for the Back item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem back;

JavaScript synopsis

menu.**back**

C synopsis

PODSMenuItem* **PODSgetBack(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Back item in the M-Business Client menus.

bookmarkManager

Accesses the object for the Bookmark Manager item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem bookmarkManager;

JavaScript synopsis

menu.**bookmarkManager**

C synopsis

PODSMenuItem* **PODSgetBookmarkManager(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Bookmark Manager item in the M-Business Client menus.

cacheManager

Accesses the object for the Online Cache Manager item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem cacheManager;

JavaScript synopsis

menu.**cacheManager**

C synopsis

PODSMenuItem* **PODSgetCacheManager(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Online Cache Manager item in the M-Business Client menus.

channelManager

Accesses the object for the Channel Manager item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem channelManager;

JavaScript synopsis

menu.**channelManager**

C synopsis

PODSMenuItem* **PODSgetChannelManager(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Cache Manager item in the M-Business Client menus.

copy

Accesses the object for the Copy item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem copy;

JavaScript synopsis

menu.**copy**

C synopsis

PODSMenuItem* **PODSgetCopy(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Copy item in the M-Business Client menus.

cut

Accesses the object for the Cut item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem cut;

JavaScript synopsis

menu.**cut**

C synopsis

PODSMenuItem* **PODSgetCut(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Cut item in the M-Business Client menus.

exit

Accesses the object for the Exit item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem exit;

JavaScript synopsis

menu.**exit**

C synopsis

PODSMenuItem* **PODSgetExit(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Exit item in the M-Business Client menus.

find

Accesses the object for the Find item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem find;

JavaScript synopsis

menu.**find**

C synopsis

PODSMenuItem* **PODSgetFind(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Find item in the M-Business Client menus.

findNext

Accesses the object for the Find Next item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem findNext;

JavaScript synopsis

menu.**findNext**

C synopsis

PODSMenuItem* **PODSgetFindNext(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Find Next item in the M-Business Client menus.

findPrevious

Accesses the object for the Find Previous item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem findPrevious;

JavaScript synopsis

menu.**findPrevious**

C synopsis

PODSMenuItem* **PODSgetFindPrevious(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Find Previous item in the M-Business Client menus.

formManager

Accesses the object for the Forms Manager item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem formManager;

JavaScript synopsis

menu.**formManager**

C synopsis

PODSMenuItem* **PODSgetFormManager(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Forms Manager item in the M-Business Client menus.

forward

Accesses the object for the Forward item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem forward;

JavaScript synopsis

menu.**forward**

C synopsis

PODSMenuItem* **PODSgetForward(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Forward item in the M-Business Client menus.

fullScreen

Accesses the object for the Full Screen item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem fullScreen;

JavaScript synopsis

menu.**fullScreen**

C synopsis

PODSMenuItem* **PODSgetFullScreen(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Full Screen item in the M-Business Client menus.

goHome

Accesses the object for the Home item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem goHome;

JavaScript synopsis

menu.**goHome**

C synopsis

PODSMenuItem* **PODSgetGoHome(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Home item in the M-Business Client menus.

help

Accesses the object for the Help item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem help;

JavaScript synopsis

menu.**help**

C synopsis

PODSMenuItem* **PODSgetHelp(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Help item in the M-Business Client menus.

isAvail()

Reflects the availability of the specified M-Business Client menu — whether it is present on the current platform and has not been removed.

Interface

PODSMenu

IDL definition

PODSBoolean isAvail();

JavaScript synopsis

menu.**help**

C synopsis

PODSBoolean* **PODSisAvail(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

PODS_TRUE if *menu* is currently available on the current device platform; PODS_FALSE otherwise.

label

Returns or sets the label that M-Business Client displays for the specified menu on the current platform.

Interface

`PODSMenuItem`

IDL definition

attribute PODSString label;

JavaScript synopsis

menuitem.**label**

menuitem.**label** = *bool*

C synopsis

PODSString **PODSgetLabel(**PODSMenu* *menu***);**

void **PODSsetLabel(**PODSMenu* *menu*, PODSString *labeltext***);**

Parameters

- ***menu*** The `PODSMenu` object.
- ***labeltext*** The text string of the label for top-level menu access.

Returns

- **Getter:** The text string of the label for the menu.
- **Setter:** None

openPage

Accesses the object for the Open Page item in the M-Business Client menus.

Interface

`PODSMenu`

IDL definition

readonly attribute PODSMenuItem openPage;

JavaScript synopsis

menu.**openPage**

C synopsis

PODSMenuItem* **PODSgetOpenPage(**PODSMenu* *menu***);**

Parameters

- ***menu*** The `PODSMenu` object.

Returns

The object for the Open Page item in the M-Business Client menus.

pageOption

Accesses the object for the Page Options item in the M-Business Client menus. On some platforms, this is called Preferences, or just Options.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem pageOption;

JavaScript synopsis

menu.**pageOption**

C synopsis

PODSMenuItem* **PODSgetPageOption(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Page Options item in the M-Business Client menus.

paste

Accesses the object for the Paste item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem paste;

JavaScript synopsis

menu.**paste**

C synopsis

PODSMenuItem* **PODSgetPaste(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Paste item in the M-Business Client menus.

reloadPage

Accesses the object for the Reload Page item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem reloadPage;

JavaScript synopsis

menu.**reloadPage**

C synopsis

PODSMenuItem* **PODSgetReloadPage(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Reload Page item in the M-Business Client menus.

remove( )

Removes the specified menu from the M-Business Client main menu. The menu is automatically restored in the user's next M-Business Client session, but cannot be restored in the current session.

Interface

PODSMenu

IDL definition

void remove();

JavaScript synopsis

menuitem.**remove()**

C synopsis

void **PODSremove(**PODSMenu* *menu***);**

void **PODSsetEnabled(**PODSMenu* *menu*, PODSBoolean *bool***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

None

selectAll

Accesses the object for the Select All item in the M-Business Client menus.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem selectAll;

JavaScript synopsis

menu.**selectAll**

C synopsis

PODSMenuItem* **PODSgetSelectAll(**PODSMenu* *menu***);**

Parameters

- ***menu*** The PODSMenu object.

Returns

The object for the Select All item in the M-Business Client menus.

serverOption

Accesses the object for the Server Options item in the M-Business Client menus. On some platforms, this is called Server Preferences.

Interface

PODSMenu

IDL definition

readonly attribute PODSMenuItem serverOption;

JavaScript synopsis

menu.**serverOption**

C synopsis

PODSMenuItem* **PODSgetServerOption(**PODSMenu* *menu***);**

Parameters

- ***menu*** The `PODSMenu` object.

Returns

The object for the Server Options item in the M-Business Client menus.

syncAll

Accesses the object for the Sync All item in the M-Business Client menus. On some platforms, this is just called Sync.

Interface

`PODSMenu`

IDL definition

readonly attribute PODSMenuItem syncAll;

JavaScript synopsis

menu.**syncAll**

C synopsis

PODSMenuItem* **PODSgetSyncAll(**PODSMenu* *menu*);**

Parameters

- ***menu*** The `PODSMenu` object.

Returns

The object for the Sync All item in the M-Business Client menus.

workOffline

Accesses the object for the Work Offline item in the M-Business Client menus.

Interface

`PODSMenu`

IDL definition

readonly attribute PODSMenuItem workOffline;

JavaScript synopsis

menu.**workOffline**

C synopsis

PODSMenuItem* **PODSgetWorkOffline(**PODSMenu* *menu*);**

Parameters

- ***menu*** The `PODSMenu` object.

Returns

The object for the Work Offline item in the M-Business Client menus.

workOnline

Accesses the object for the Work Online item in the M-Business Client menus.

Interface

`PODSMenu`

IDL definition

readonly attribute PODSMenuItem workOnline;

JavaScript synopsis

menu.**workOnline**

C synopsis

PODSMenuItem* **PODSgetWorkOnline(**PODSMenu* *menu***);**

Parameters

- ***menu*** The `PODSMenu` object.

Returns

The object for the Work Online item in the M-Business Client menus.

PODSMenuItem object

Each of the `PODSMenuItem` attributes and methods work with a menu item object that has been accessed through a `PODSMenu` attribute.

Inherits from: `PODSObject`

- **Accessed by:** `PODSMenu` object attribute for the specified menu item
- **Available to:** C, JavaScript

Summary of PODSMenuItem attributes and methods

Description	Attributes and methods
Menu item attributes	“enabled” on page 284 “label” on page 285
Menu item availability	“isAvail( )” on page 285 “remove( )” on page 286

enabled

Determines whether the specified M-Business Client menu item is enabled or disabled.

> **Note**
> On some menu items, such as Work Online/Offline, M-Business Client will override any changes in the menu item's enabled status that conflict with the device state. This will happen the first time that the user displays the menu containing that menu item, so the user will never see the conflicting setting.

Interface

`PODSMenuItem`

IDL definition

attribute PODSBoolean enabled;

JavaScript synopsis

menuitem.**enabled**

menuitem.**enabled** = *bool*

C synopsis

PODSBoolean **PODSgetEnabled(**PODSMenuItem* *menuitem*);

void **PODSsetEnabled(**PODSMenuItem* *menuitem*, PODSBoolean *bool***);**

Parameters

- ***menuitem*** The `PODSMenuItem` object.
- ***bool*** A boolean value: `PODS_TRUE` to enable the menu item; `PODS_FALSE` to disable it.

Returns

- **Getter:** `PODS_TRUE` if the menu item is enabled; `PODS_FALSE` otherwise.
- **Setter:** None

isAvail( )

Reflects the availability of the specified M-Business Client menu item — whether it is present — on the current device platform.

Interface

`PODSMenuItem`

IDL definition

PODSBoolean isAvail();

JavaScript synopsis

menuitem.**isAvail()**

C synopsis

PODSBoolean **PODSgetEnabled(**PODSMenuItem* *menuitem***);**

Parameters

- ***menuitem*** The `PODSMenuItem` object.

Returns

`PODS_TRUE` if the menu item is available on the current device platform; `PODS_FALSE` otherwise.

label

The label that M-Business Client displays for the specified menu item.

Interface

`PODSMenuItem`

IDL definition

attribute PODSString label;

JavaScript synopsis

menuitem.**label**

menuitem.**label** = *labeltext*

C synopsis

PODSString **PODSgetLabel(**PODSMenuItem* *menuitem*);

void **PODSsetLabel(**PODSMenuItem* *menuitem*, PODSString *labeltext*);

Parameters

- ***menuitem*** The `PODSMenuItem` object.
- ***labeltext*** The text string of the label for the menu item.

Returns

- **Getter:** The text string of the label for the menu item.
- **Setter:** None

remove( )

Removes the specified M-Business Client menu item that would otherwise be available on the current device platform. Removed menu items are automatically restored in the user's next M-Business Client session, but cannot be restored in the current session. "isAvail( )" on page 285 returns `PODS_FALSE` for removed menu items.

Interface

`PODSMenuItem`

IDL definition

void remove();

JavaScript synopsis

menuitem.**remove()**

C synopsis

void **PODSremove(**PODSMenuItem* *menuitem*);

Parameters

- ***menuitem*** The `PODSMenuItem` object.

Returns

None

CHAPTER 11

PODS miscellaneous objects

Contents

PODSScreen object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSWindow` object's "screen" on page 248
- **Available to:** C, JavaScript

The `PODSScreen` attributes and methods access information about the screen on which M-Business Client is running.

Summary of PODSScreen attributes and methods

Description	Attributes and methods
Getting available screen data	"availHeight" on page 288 "availLeft" on page 289 "availTop" on page 289 "availWidth" on page 290
Getting screen parameters	"colorDepth" on page 291 "height" on page 291 "pixelDepth" on page 292 "width" on page 292

availHeight

The available height for the `PODSScreen` object; the height of the screen, in pixels, minus permanent or semipermanent user interface features displayed by the operating system.

Interface

`PODSScreen`

IDL definition

read-only attribute PODSInt32 availHeight;

JavaScript synopsis

screen.**availHeight**

C synopsis

PODSInt32 **PODSgetAvailHeight(**PODSScreen* *screen*);

Parameters

- ***screen*** The `PODSScreen` object.

Returns

The available height for the `PODSScreen` object.

See also

"height" on page 291

availLeft

The available left for the `PODSScreen` object; the x-coordinate of the first pixel that is not allocated to permanent or semipermanent user interface features.

Interface

`PODSScreen`

IDL definition

read-only attribute PODSInt32 availLeft;

JavaScript synopsis

screen.**availLeft**

C synopsis

PODSInt32 **PODSgetAvailLeft(**PODSScreen* *screen***);**

Parameters

- ***screen*** The `PODSScreen` object.

Returns

The available left for the `PODSScreen` object.

See also

"availTop" on page 289, "width" on page 292

availTop

The available top for the `PODSScreen` object; the y-coordinate of the first pixel that is not allocated to permanent or semipermanent user interface features.

Interface

`PODSScreen`

IDL definition

read-only attribute PODSInt32 availTop;

JavaScript synopsis

screen.**availTop**

C synopsis

PODSInt32 **PODSgetAvailTop(**PODSScreen* *screen*);

Parameters

- ***screen*** The `PODSScreen` object.

Returns

The available top for the `PODSScreen` object.

See also

"availHeight" on page 288

availWidth

The available width for the `PODSScreen` object; the width of the screen, in pixels, minus permanent or semipermanent user interface features displayed by the operating system.

Interface

`PODSScreen`

IDL definition

read-only attribute PODSInt32 availWidth;

JavaScript synopsis

screen.**availWidth**

C synopsis

PODSInt32 **PODSgetAvailWidth(**PODSScreen* *screen*);

Parameters

- ***screen*** The `PODSScreen` object.

Returns

The available width for the `PODSScreen` object.

See also

"width" on page 292

colorDepth

The color bit depth used to display images on the `PODSScreen` object.

Interface

`PODSScreen`

IDL definition

read-only attribute PODSInt32 colorDepth;

JavaScript synopsis

screen.**colorDepth**

C synopsis

PODSInt32 **PODSgetColorDepth(**PODSScreen* *screen*);

Parameters

- ***screen*** The `PODSScreen` object.

Returns

The color depth for the `PODSScreen` object.

Remarks

The color depth is the bit depth of the color palette in bits per pixel, if a color palette is in use. Otherwise, this property is derived from `pixelDepth`.

See also

"pixelDepth" on page 292

height

The height for the `PODSScreen` object; the display screen height, in pixels.

Interface

`PODSScreen`

IDL definition

read-only attribute PODSInt32 height;

JavaScript synopsis

screen.**height**

C synopsis

PODSInt32 **PODSgetHeight(**PODSScreen* *screen*);

Parameters

- ***screen*** The `PODSScreen` object.

Returns

The height for the `PODSScreen` object.

See also

"availHeight" on page 288

pixelDepth

The pixel depth for the `PODSScreen` object; the display screen color resolution, in bits per pixel.

Interface

`PODSScreen`

IDL definition

read-only attribute PODSInt32 pixelDepth;

JavaScript synopsis

screen.**pixelDepth**

C synopsis

PODSInt32 **PODSgetPixelDepth(**PODSScreen* *screen***);**

Parameters

- ***screen*** The `PODSScreen` object.

Returns

The pixel depth for the `PODSScreen` object.

See also

"colorDepth" on page 291

width

The width for the `PODSScreen` object.

Interface

`PODSScreen`

IDL definition

read-only attribute PODSInt32 width;

JavaScript synopsis

screen.**width**

C synopsis

PODSInt32 **PODSgetWidth(**PODSScreen* *screen*);

Parameters

- ***screen*** The `PODSScreen` object.

Returns

The width for the `PODSScreen` object.

Remarks

Width is the display screen width, in pixels.

See also

"availWidth" on page 290

PODSPrefs object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSAvantGo` object's "preferences" on page 91
- **Available to:** C (fully), JavaScript (selectively)

The `PODSPrefs` attributes and methods allow you to get and set M-Business Client's user preference values. Typically users set M-Business Client preferences through its **Options** menu. Your application should respect and adhere to the values that the user has set. Changing M-Business Client preferences is allowed, but you should do so only when absolutely necessary.

PODSPrefs key values

The `PODSPrefs` *key* values, defined in the *pods.h* file, are listed in the table below. These *key* values are used in all `PODSPrefs` methods.

Table 1. PODSPrefs key values from pods.h file

Constant	String value
`PREFS_DOWNLOADIMAGES`	`"DownloadImages"`
`PREFS_ONLINEIMAGEBITDEPTH`	`"OnlineImageBitDepth"`
`PREFS_MAXIMUMCACHESIZE`	`"MaximumCacheSize"`
`PREFS_SHOWIMAGES`	`"ShowImages"`
`PREFS_SHOWDETAILEDRESULTS`	`"ShowDetailedResults"`
`PREFS_ALWAYSSHOWOPENWINDOW`	`"AlwaysShowOpenPageWindow"`
`PREFS_ADVANCEDPAD`	`"AdvancedPad"`
`PREFS_ENABLEHARDKEYS`	`"EnableHardKeys"`
`PREFS_SHOWTABLES`	`"ShowTables"`
`PREFS_AUTOCONNECT`	`"AutoConnect"`
`PREFS_ENABLEJS`	`"EnableJS"`
`PREFS_SCANNERINITIALLYENABLED`	`"ScannerPrevEnabled"`
`PREFS_LASTSERVERID`	`"LastServerUid"`
`PREFS_SEQUENCE_COOKIE`	`"LastSequenceCookie"`
`PREFS_DRAGACTION`	`"DragAction"`

Constant	String value
PREFS_MAXONLINEREQUESTSIZE	"MaxOnlineRequestSize"
PREFS_REFRESH_EXPIRED_PAGES	"RefreshExpiredPages"
PREFS_SHOW_JS_ERRORS	"ShowJSErrors"
PREFS_FULLSCREEN	"FullScreen"
PREFS_FONTMINSIZE	"FontMinSize"
PREFS_FLUSHSUBMISSIONSONLINE	"FlushSubmissionsOnline"
PREFS_SAVESUBMISSIONONONLINEERR	"SaveSubmissionOnOnlineError"
PREFS_DISABLERIGHTCLICKPOPUP	"DisableRightClickPopup"

Summary of PODSPrefs attributes and methods

Table 2. Summary of PODSPrefs attributes and methods

Description	Attributes and methods
Accessing integer values	"getStringValueForKey( )" on page 298 "setStringValueForKey( )" on page 302 "getInt32ValueForKey( )" on page 297 "setInt32ValueForKey( )" on page 301 "getUInt32ValueForKey( )" on page 298 "setUInt32ValueForKey( )" on page 301
Accessing boolean values	"getBoolValueForKey( )" on page 295 "setBoolValueForKey( )" on page 299
Accessing byte values	"getBytesForKey( )" on page 296 "setBytesForKey( )" on page 300

getBoolValueForKey( )

Gets the boolean value for the specified preference key.

Interface

`PODSPrefs`

IDL definition

PODSBoolean getBoolValueForKey(PODSString *key*);

JavaScript synopsis

avantgo.*preferences*.**getBoolValueForKey(***key***)**

C synopsis

PODSBoolean **PODSgetBoolValueForKey(**
  PODSPrefs* *preferences*,
  PODSString *key*
);

Parameters

- ***preferences*** The `PODSPrefs` object.
- ***key*** [in] Preference key value. See "PODSPrefs key values" on page 294.

Returns

`PODS_TRUE`

If the boolean value for the key is set to `PODS_TRUE`.

`PODS_FALSE`

Otherwise.

See also

"setBoolValueForKey( )" on page 299

getBytesForKey( )

Gets the binary data bytes for the specified preference key. The binary data bytes are set by "setBytesForKey( )" on page 300.

Interface

`PODSPrefs`

IDL definition

nometadata PODSUInt8 getBytesForKey(
  PODSString *key*,
  PODSInt32 *length*
);

JavaScript synopsis

Not applicable

C synopsis

PODSBoolean **PODSgetBoolValueForKey(**
  PODSPrefs* *preferences*,


```
  PODSString key
);
```

Parameters

- ***preferences*** The `PODSPrefs` object.
- ***key*** [in] Preference key value. See "PODSPrefs key values" on page 294.
- ***length*** [in] Length of binary data, in bytes.

Returns

Binary data bytes for specified preference key value.

Remarks

The data bytes associated with a preference key value may be used to store any binary data that needs to be associated with the key.

See also

"setBytesForKey( )" on page 300

getInt32ValueForKey( )

Gets the signed integer value for the specified preference key.

Interface

`PODSPrefs`

IDL definition

PODSInt32 getInt32ValueForKey(PODSString *key*);

JavaScript synopsis

avantgo.*preferences*.**getInt32ValueForKey(***key***)**

C synopsis

```
PODSInt32 PODSgetInt32ValueForKey(
  PODSPrefs* preferences,
  PODSString key
);
```

Parameters

- ***preferences*** The `PODSPrefs` object.
- ***key*** [in] Preference key value. See "PODSPrefs key values" on page 294.

Returns

Signed integer value for the specified preference key.

See also

"setInt32ValueForKey( )" on page 301

getUInt32ValueForKey( )

Gets the unsigned integer value for the specified preference key.

Interface

`PODSPrefs`

IDL definition

PODSUInt32 getUInt32ValueForKey(PODSString *key*);

JavaScript synopsis

avantgo.*preferences*.**getUInt32ValueForKey(***key***)**

C synopsis

```
PODSUInt32 PODSgetUInt32ValueForKey(
  PODSPrefs* preferences,
  PODSString key
);
```

Parameters

- ***preferences*** The `PODSPrefs` object.
- ***key*** [in] Preference key value. See "PODSPrefs key values" on page 294.

Returns

Unsigned integer value for the specified preference key.

See also

"setUInt32ValueForKey( )" on page 301

getStringValueForKey( )

Gets the string value for the specified preference key.

Interface

`PODSPrefs`

IDL definition

PODSString getStringValueForKey(PODSString *key*);

JavaScript synopsis

avantgo.*preferences*.**getStringValueForKey(***key***)**

C synopsis

```
PODSString PODSgetStringValueForKey(
  PODSPrefs* preferences,
  PODSString key
);
```

Parameters

- ***preferences*** The `PODSPrefs` object.
- ***key*** [in] Preference key value. See "PODSPrefs key values" on page 294.

Returns

String value for the specified preference key.

See also

"getStringValueForKey( )" on page 298

setBoolValueForKey( )

Sets the boolean value for the specified preference key.

Interface

`PODSPrefs`

IDL definition

```
void setBoolValueForKey(
  PODSString key,
  PODSBoolean bool
);
```

JavaScript synopsis

avantgo.*preferences*.**setBoolValueForKey(***key*, *bool***)**

C synopsis

```
void PODSsetBoolValueForKey(
  PODSPrefs* preferences,
  PODSString key,
  PODSBoolean bool
);
```

Parameters

- ***preferences*** The `PODSPrefs` object.
- ***key*** [in] Preference key value. See "PODSPrefs key values" on page 294.
- ***bool*** [in] Boolean value to set: `PODS_TRUE` or `PODS_FALSE`.

Returns

None

See also

"getBoolValueForKey( )" on page 295

setBytesForKey( )

Sets the binary data bytes for the specified preference key.

Interface

`PODSPrefs`

IDL definition

```
nometadata void setBytesForKey(
  PODSString key,
  PODSUInt8 data,
  PODSUInt32 length
);
```

JavaScript synopsis

Not applicable

C synopsis

```
void PODSsetBytesForKey(
  PODSPrefs* preferences,
  PODSString key,
  PODSUInt8* data,
  PODSUInt32 length
);
```

Parameters

- ***preferences*** The `PODSPrefs` object.
- ***key*** [in] Preference key value. See "PODSPrefs key values" on page 294.
- ***data*** [in] The binary data to set for the key value.
- ***length*** [in] Length of binary data, in bytes.

Returns

None

Remarks

The data bytes associated with a preference key value may be used to store any binary data that needs to be associated with the key.

See also

"getBytesForKey( )" on page 296

setInt32ValueForKey( )

Sets the signed integer value for the specified preference key.

Interface

`PODSPrefs`

IDL definition

```
void setInt32ValueForKey(
  PODSString key,
  PODSUInt32 value
);
```

JavaScript synopsis

avantgo.*preferences*.**setInt32ValueForKey(***key***,** *value***)**

C synopsis

```
void PODSsetInt32ValueForKey(
  PODSPrefs* preferences,
  PODSString key,
  PODSUInt32 value
);
```

Parameters

- ***preferences*** The `PODSPrefs` object.
- ***key*** [in] Preference key value. See "PODSPrefs key values" on page 294.
- ***value*** [in] Signed integer value to set.

Returns

None

See also

"getInt32ValueForKey( )" on page 297

setUInt32ValueForKey( )

Sets the unsigned integer value for the specified preference key.

Interface

`PODSPrefs`

IDL definition

```
void setUInt32ValueForKey(
  PODSString key,
  PODSUInt32 value
);
```


JavaScript synopsis

avantgo.*preferences*.**setUInt32ValueForKey(***key***,** *value***)**

C synopsis

```
void PODSsetUInt32ValueForKey(
  PODSPrefs* preferences,
  PODSString key,
  PODSUInt32 value
);
```

Parameters

- ***preferences*** The `PODSPrefs` object.
- ***key*** [in] Preference key value. See "PODSPrefs key values" on page 294.
- ***value*** [in] Unsigned integer value to set.

Returns

None

See also

"getUInt32ValueForKey( )" on page 298

setStringValueForKey( )

Sets the string value for the specified preference key.

Interface

`PODSPrefs`

IDL definition

```
PODSBoolean setStringValueForKey(
  PODSString key,
  PODSString valueStr
);
```

JavaScript synopsis

avantgo.*preferences*.**setStringValueForKey(***key***,** *valueStr***)**

C synopsis

```
PODSBoolean PODSsetStringValueForKey(
  PODSPrefs* preferences,
  PODSString key,
  PODSString valueStr
);
```

Parameters

- ***preferences*** The `PODSPrefs` object.

- ***key*** [in] Preference key value. See "PODSPrefs key values" on page 294.
- ***valueStr*** [in] Value to which preference is to be set.

Returns

`PODS_TRUE`

If the preference is successfully set.

`PODS_FALSE`

Otherwise.

See also

"getStringValueForKey( )" on page 298

PODSNavigator object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSWindow` object's "navigator" on page 246
- **Available to:** C, JavaScript

The `PODSNavigator` attributes and methods return information about the M-Business Client application.

Summary of PODSNavigator attributes and methods

Description	Attributes and methods
Getting application (browser) data	"appCodeName" on page 304 "appName" on page 305 "appVersion" on page 305
Getting environment data	"platform( )" on page 306 "javaEnabled( )" on page 306

appCodeName

The application code name; the code name of the browser.

Interface

`PODSNavigator`

IDL definition

readonly attribute PODSString appCodeName;

JavaScript synopsis

nav.**appCodeName**

C synopsis

PODSString **PODSgetAppCodeName(**PODSNavigator* *nav*);

Parameters

- ***nav*** The `PODSNavigator` object.

Returns

The application code name.

See also

"appName" on page 305, "appVersion" on page 305

appName

The application name; the name of the browser.

Interface

`PODSNavigator`

IDL definition

readonly attribute PODSString appName;

JavaScript synopsis

nav.**appName**

C synopsis

PODSString **PODSgetAppName(**PODSNavigator* *nav***);**

Parameters

- ***nav*** The `PODSNavigator` object.

Returns

The application name.

See also

"appCodeName" on page 304, "appVersion" on page 305

appVersion

Gets the application version, contains version information for the browser.

Interface

`PODSNavigator`

IDL definition

readonly attribute PODSString appVersion;

JavaScript synopsis

nav.**appVersion**

C synopsis

PODSString **PODSgetAppVersion(**PODSNavigator* *nav***);**

Parameters

- ***nav*** The `PODSNavigator` object.

Returns

The application version.

See also

"appName" on page 305, "appCodeName" on page 304

javaEnabled( )

The value of `javaEnabled` for the `PODSNavigator` object.

Interface

`PODSNavigator`

IDL definition

PODSBoolean javaEnabled( );

JavaScript synopsis

nav.**javaEnabled( )**

C synopsis

PODSBoolean **PODSjavaEnabled(**PODSNavigator* *nav***);**

Parameters

nav

The `PODSNavigator` object.

Returns

`PODS_FALSE` In all cases. (Currently M-Business Client is not Java enabled.)

See also

"platform( )" on page 306

platform( )

The platform; the machine type for which the browser code was compiled.

Interface

`PODSNavigator`

IDL definition

read-only attribute PODSString platform;

JavaScript synopsis

nav.**platform**

C synopsis

PODSString **PODSgetPlatform(**PODSNavigator* *nav***);**

Parameters

- ***nav*** The `PODSNavigator` object.

Returns

The platform.

See also

"javaEnabled( )" on page 306

PODSSymbolScanner object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSAvantGo` object's "createObject( )" on page 85, passing `"avantgo.symbolScanner"` as the object name
- **Available to:** C, JavaScript

> **Note**
> The `PODSSymbolScanner` interface implements a Symbol Technologies API for the Palm OS and WinCE platforms. You cannot use `PODSSymbolScanner` on other platforms.

The `PODSSymbolScanner` object provides the API for working with a Symbol Technologies bar-code scanner device. Your POD might allow users to update information in a POD-specific database by scanning in that new information. The `PODSSymbolScanner` object provides a way to do this.

You access the `PODSSymbolScanner` object by using the `PODSWindow` method "createObject( )" on page 85. Because the `PODSWindow` object creates the `PODSSymbolScanner` object, you do not have to free the memory associated with it.

The `PODSSymbolScanner` object's API is essentially a cover for the Scan Manager API provided by Symbol Technologies. Refer to the Symbol Technologies documentation for complete descriptions of how to use the Scan Manager.

Opening the Scan Manager library

There is one significant difference between the Scan Manager API and the `PODSSymbolScanner` API. The Scan Manager API requires you to open the Scan Manager library before you can make any other call to the library. `PODSSymbolScanner` opens the library for you whenever you enable a piece of the scanning hardware (that is, the aimer, the LED, or the scanner itself) and the library is closed.

Any time you disable the scanner, LED, or aimer, `PODSSymbolScanner` checks to see if any of the other scanning pieces are still enabled. If not, it closes the library for you. Closing the library when it is not in use prolongs battery life.

Calls that do not actually require use of the scanning hardware (such as the calls that retrieve version numbers) open the library, retrieve the requested information, and close the library when finished. The majority of `PODSSymbolScanner` calls, however, simply return `NULL` or -1 if the Scan Manager library is not open. If you want to explicitly control the opening and closing of the library, you can use `PODSSetEnabled()`.

PODSSymbolScanner calls

The table below lists each Scan Manager function and its corresponding `PODSSymbolScanner` syntax.

In C, each `PODSSymbolScanner` method takes one more argument than its Scan Manager counterpart and that is the pointer to the `PODSSymbolScanner` object, which is always the first parameter.

Table 3. Symbol Technologies methods, PODSSYMBOLSCANNER calls

Symbol Technologies method	M-Business JavaScript synopsis	M-Business C synopsis
Scan**OpenDecoder**	avantgo.symbolScanner. **Enabled** = true	PODSInt16 **PODSsetEnabled**(PODSSymbolScanner* *scanobj*, PODS_TRUE);
Scan**CloseDecoder**	avantgo.symbolScanner. **Enabled** = false	PODSInt16 **PODSsetEnabled**(PODSSymbolScanner* *scanobj*, PODS_FALSE)
(Scan**OpenDecoder** - query decoder's state; has no separate command)	avantgo.symbolScanner. **Enabled**	PODSBoolean **PODSgetEnabled**(PODSSymbolScanner* *scanobj*);
Scan**CmdSendParams**	avantgo.symbolScanner. **SendParams**()	PODSInt16 **PODSsendParams**(PODSSymbolScanner* *scanobj*, PODSSymbolBeepType *beepType*);
Scan**CmdGetAllParams**	avantgo.symbolScanner. **GetAllParams**()	PODSString **PODSgetAllParams**(PODSSymbolScanner* *scanobj*);
Scan**CmdAimOn**	avantgo.symbolScanner. **AimEnabled** = true	PODSInt16 **PODSsetAimEnabled**(PODSSymbolScanner* *scanobj*, PODS_TRUE);
Scan**CmdAimOff**	avantgo.symbolScanner. **AimEnabled** = false	PODSInt16 **PODSsetAimEnabled**(PODSSymbolScanner* *scanobj*, PODS_FALSE);
Scan**CmdBeep**	avantgo.symbolScanner. **Beep**()	PODSInt16 **PODSbeep**(PODSSymbolScanner* C, PODSSymbolBeepType *beepType*);

Symbol Technologies method	M-Business JavaScript synopsis	M-Business C synopsis
Scan**GetDecodedData**	avantgo.symbolScanner. **ScanData** avantgo.symbolScanner. **ScanType** avantgo.symbolScanner. **ScanStatus**	PODSString **PODSgetScanData**(PODSSymbolScanner* *scanobj*); /* returns ptr->data */ PODSInt16 **PODSgetScanType**(PODSSymbolScanner* *scanobj*); /* returns ptr->type */ PODSInt16 **PODSgetScanStatus**(PODSSymbolScanner* *scanobj*); /* returns value from ScanGetDecodedData */
Scan**CmdLedOn**	avantgo.symbolScanner. **LedEnabled** = true	PODSInt16 **PODSsetLedEnabled**(PODSSymbolScanner* *scanobj*, PODS_TRUE)
Scan**CmdLedOff**	avantgo.symbolScanner. **LedEnabled** = false	PODSInt16 **PODSsetLedEnabled**(PODSSymbolScanner* *scanobj*, PODS_FALSE)
Scan**CmdParamDefaults**	avantgo.symbolScanner. **SetDefaultParams**	PODSInt16 **PODSsetDefaultParams**(PODSSymbolScanner* *scanobj*);
Scan**GetDecoder Version**	avantgo.symbolScanner. **DecoderVersion**	PODSString **PODSgetDecoderVersion**(PODSSymbolScanner* *scanobj*);
Scan**GetScanManager Version**	avantgo.symbolScanner. **ScanMgrVersion**	PODSString **PODSgetScanMgrVersion**(PODSSymbolScanner* *scanobj*);
Scan**GetScanPort DriverVersion**	avantgo.symbolScanner. **PortDriverVersion**	PODSString **PODSgetPortDriverVersion**(PODSSymbolScanner* *scanobj*);
Scan**CmdScanEnable**	avantgo.symbolScanner. **ScanEnabled** = true	PODSInt16 **PODSsetScanEnabled**(PODSSymbolScanner* *scanobj*, PODS_TRUE);

Symbol Technologies method	M-Business JavaScript synopsis	M-Business C synopsis
Scan**CmdScanDisable**	avantgo.symbolScanner. **ScanEnabled** = false	PODSInt16 **PODSsetScanEnabled**(PODSSymbolScanner* *scanobj*, PODS_FALSE);
(Scan**CmdScanEnable** - query scan enabled state; has no separate command)	avantgo.symbolScanner. **ScanEnabled**	PODSBoolean **PODSgetScanEnabled**(PODSSymbolScanner* *scanobj*);
Scan**CmdStartDecode**	avantgo.symbolScanner. **StartDecode**()	PODSInt16 **PODSstartDecode**(PODSSymbolScanner* *scanobj*);
Scan**CmdStopDecode**	avantgo.symbolScanner. **StopDecode**()	PODSInt16 **PODSstopDecode**(PODSSymbolScanner* *scanobj*);
Scan**SetBarcode Enabled**	avantgo.symbolScanner. **SetBarcodeEnabled**()	PODSInt16 **PODSsetBarcodeEnabled**(PODSSymbolScanner* *scanobj*, PODSSymbolBarcodeType barCode, PODSBoolean *enable*);
Scan**GetBarcode Enabled**	avantgo.symbolScanner. **GetBarcodeEnabled**()	PODSBoolean **PODSgetBarcodeEnabled**(PODSSymbolScanner* *scanobj*, PODSSymbolBarcodeType *barCode*);
Scan**SetTransmit CheckDigit**	avantgo.symbolScanner. **SetTransmitCheckDigit Enabled**()	PODSInt16 **PODSsetTransmitCheckDigit Enabled**(PODSSymbolScanner* *scanobj*, PODSSymbolBarcodeType *barCode*, PODSBoolean *enable*);
Scan**GetTransmit CheckDigit**	avantgo.symbolScanner. **GetTransmitCheckDigit Enabled**()	PODSBoolean **PODSgetTransmitCheckDigit Enabled**(PODSSymbolScanner* *scanobj*, PODSSymbolBarcodeType *barCode*);

Symbol Technologies method	M-Business JavaScript synopsis	M-Business C synopsis
Scan**SetConvert**	avantgo.symbolScanner. **SetConversion** **Enabled**()	PODSInt16 **PODSsetConversion** **Enabled**(PODSSymbolScanner* *scanobj*, PODSSymbolConvertType *convType*, PODSBoolean *enable*);
Scan**GetConvert**	avantgo.symbolScanner. **GetConversion** **Enabled**()	PODSBoolean **PODSgetConversionEnabled**(PODSSymbolScanner* *scanobj* PODSSymbolConvertType *convType*);
Scan**SetBarcode** **Lengths**	avantgo.symbolScanner. **SetBarcodeLengths**(*lengthType*, *length1*, *length2*)	PODSInt16 **PODSsetBarcodeLengths**(PODSSymbolScanner* *scanobj*, PODSSymbolBarcodeType *barCode*, PODSUInt16 *lengthType*, PODSUInt16 *length1*, PODSUInt16 *length2*);
Scan**GetBarcode** **Lengths**	avantgo.symbolScanner. **GetBarcodeLength** **LenType**() avantgo.symbolScanner. **GetBarcodeLength** **Len1**() avantgo.symbolScanner. **GetBarcodeLength** **Len2**()	PODSInt16 **PODSgetBarcodeLengthLen** **Type**(PODSSymbolScanner* *scanobj*, PODSSymbolBarcodeType *barCode*); PODSInt16 **PODSgetBarcodeLengthLen1**(PODSSymbolScanner* *scanobj*, PODSSymbolBarcodeType *barCode*); PODSInt16 **PODSgetBarcodeLengthLen2**(PODSSymbolScanner* *scanobj*, PODSSymbolBarcodeType *barCode*);
Scan**SetEanZeroExtend**	avantgo.symbolScanner. **SetEanZeroExtend**()	PODSInt16 **PODSsetEanZeroExtend**(PODSSymbolScanner* *scanobj*, PODSBoolean *enable*);
Scan**GetEanZeroExtend**	avantgo.symbolScanner. **GetEanZeroExtend**()	PODSInt16 **PODSgetEanZeroExtend**(PODSSymbolScanner* *scanobj*);

Symbol Technologies method	M-Business JavaScript synopsis	M-Business C synopsis
Scan**SetDecodeUpcEanSupplementals**	avantgo.symbolScanner.**DecodeUpcEanSupplementals**	PODSInt16 **PODSsetDecodeUpcEanSupplementals**(PODSSymbolScanner* *scanobj*, PODSUInt16 *supplementals*);
Scan**GetDecodeUpcEanSupplementals**	avantgo.symbolScanner.**DecodeUpcEanSupplementals**	PODSInt16 **PODSgetDecodeUpcEanSupplementals**(PODSSymbolScanner* *scanobj*);
Scan**SetDecodeUpcEanRedundancy**	avantgo.symbolScanner.**DecodeUpcEanRedundancy**	PODSInt16 **PODSsetDecodeUpcEanRedundancy**(PODSSymbolScanner* *scanobj*, PODSUInt16 *supp_redundancy*);
Scan**GetDecodeUpcEanRedundancy**	avantgo.symbolScanner.**DecodeUpcEanRedundancy**	PODSInt16 **PODSgetDecodeUpcEanRedundancy**(PODSSymbolScanner* *scanobj*);
Scan**SetUpcPreamble**	avantgo.symbolScanner.**SetUpcPreamble**()	PODSInt16 **PODSsetUpcPreamble**(PODSSymbolScanner* *scanobj*, PODSUInt16 **preamble**);
Scan**GetUpcPreamble**	avantgo.symbolScanner.**GetUpcPreamble**()	PODSInt16 **PODSgetUpcPreamble**(PODSSymbolScanner* *scanobj*,
Scan**SetUpcEanSecurityLevel**	avantgo.symbolScanner.**UpcEanSecurityLevel**	PODSInt16 **PODSsetUpcEanSecurityLevel**(PODSSymbolScanner* *scanobj*, PODSUInt16 *securityLevel*);
Scan**GetUpcEanSecurityLevel**	avantgo.symbolScanner.**UpcEanSecurityLevel**	PODSInt16 **PODSgetUpcEanSecurityLevel**(PODSSymbolScanner* *scanobj*);
Scan**SetCode39CheckDigitVerification**	avantgo.symbolScanner.**SetCode39CheckDigitVerificationEnabled**()	PODSInt16 **PODSsetCode39CheckDigitVerificationEnabled**(PODSSymbolScanner* *scanobj*, PODSBoolean *enable*);

Symbol Technologies method	M-Business JavaScript synopsis	M-Business C synopsis
Scan**GetCode39 CheckDigit Verification**	avantgo.symbolScanner. **GetCode39CheckDigit VerificationEnabled**()	PODSBoolean **PODSgetCode39CheckDigit VerificationEnabled**(PODSSymbolScanner* *scanobj*);
Scan**SetCode39Full Ascii**	avantgo.symbolScanner. **Code39FullAscii**	PODSInt16 **PODSsetCode39FullAscii**(PODSSymbolScanner* *scanobj*, PODSUInt16 *fullAscii*);
Scan**GetCode39Full Ascii**	avantgo.symbolScanner. **Code39FullAscii**	PODSInt16 **PODSgetCode39FullAscii**(PODSSymbolScanner* *scanobj*);
Scan**SetCode32Prefix**	avantgo.symbolScanner. **SetCode32Prefix**()	PODSInt16 **PODSsetCode32Prefix**(PODSSymbolScanner* *scanobj*, PODSBoolean *enable*);
Scan**GetCode32Prefix**	avantgo.symbolScanner. **GetCode32Prefix**()	PODSBoolean **PODSgetCode32Prefix**(PODSSymbolScanner* *scanobj*);
Scan**SetI2of5 CheckDigit Verification**	avantgo.symbolScanner. **I2of5CheckDigit Verification**	PODSInt16 **PODSsetI2of5CheckDigit Verification**(PODSSymbolScanner* *scanobj*, PODSUInt16 *checkDigit*);
Scan**GetI2of5 CheckDigit Verification**	avantgo.symbolScanner. **I2of5CheckDigit Verification**	PODSInt16 **PODSgetI2of5CheckDigit Verification**(PODSSymbolScanner* *scanobj*);
Scan**SetClsiEditing**	avantgo.symbolScanner. **SetClsiEditing**()	PODSInt16 **PODSsetClsiEditing**(PODSSymbolScanner* *scanobj*, PODSBoolean *enable*);
Scan**GetClsiEditing**	avantgo.symbolScanner. **GetClsiEditing**()	PODSInt16 **PODSgetClsiEditing**(PODSSymbolScanner* *scanobj*);
Scan**SetNotisEditing**	avantgo.symbolScanner. **SetNotisEditing**()	PODSInt16 **PODSsetNotisEditing**(PODSSymbolScanner* *scanobj*, PODSBoolean *enable*);

Symbol Technologies method	M-Business JavaScript synopsis	M-Business C synopsis
Scan**GetNotisEditing**	avantgo.symbolScanner. **GetNotisEditing**()	PODSInt16 **PODSgetNotisEditing**(PODSSymbolScanner* *scanobj*);
Scan**SetMsiPlessey** **CheckDigits**	avantgo.symbolScanner. **MsiPlesseyCheckDigits**	PODSInt16 **PODSsetMsiPlesseyCheckDigits**(PODSSymbolScanner* *scanobj*, PODSUInt16 *checkDigits*);
Scan**GetMsiPlessey** **CheckDigits**	avantgo.symbolScanner. **MsiPlesseyCheckDigits**	PODSInt16 **PODSgetMsiPlesseyCheckDigits**(PODSSymbolScanner* *scanobj*);
Scan**SetMsiPlessey** **CheckDigit** **Algorithm**	avantgo.symbolScanner. **SetMsiPlessey** **CheckDigitAlgorithm**	PODSInt16 **PODSsetMsiPlesseyCheck** **DigitAlgorithm**(PODSSymbolScanner* *scanobj*, PODSUInt16 *algorithm*);
Scan**GetMsiPlessey** **CheckDigit** **Algorithm**	avantgo.symbolScanner. **GetMsiPlessey** **CheckDigitAlgorithm**	PODSInt16 **PODSgetMsiPlesseyCheck** DigitAlgorithm(PODSSymbolScanner* *scanobj*);
Scan**SetScanAngle**	avantgo.symbolScanner. **Angle**	PODSInt16 **PODSsetAngle**(PODSSymbolScanner* *scanobj*, PODSUInt16 *scanAngle*);
Scan**GetAngle**	avantgo.symbolScanner. **Angle**	PODSInt16 **PODSgetAngle**(PODSSymbolScanner* *scanobj*);
Scan**SetLaserOnTime**	avantgo.symbolScanner. **LaserOnTime**	PODSInt16 **PODSsetLaserOnTime**(PODSSymbolScanner* *scanobj*, PODSUInt16 *laserOnTime*);
Scan**GetLaserOnTime**	avantgo.symbolScanner. **LaserOnTime**	PODSInt16 **PODSgetLaserOnTime**(PODSSymbolScanner* *scanobj*);
Scan**SetDecodeLed** **OnTime**	avantgo.symbolScanner. **DecodeLedOnTime**	PODSInt16 **PODSsetDecodeLedOnTime**(PODSSymbolScanner* *scanobj*, PODSUInt16 *ledOnTime*);

Symbol Technologies method	M-Business JavaScript synopsis	M-Business C synopsis
Scan**GetDecodeLed OnTime**	avantgo.symbolScanner. **DecodeLedOnTime**	PODSInt16 **PODSgetDecodeLedOnTime**(PODSSymbolScanner* *scanobj*);
Scan**SetAimDuration**	avantgo.symbolScanner. **AimDuration**	PODSInt16 **PODSsetAimDuration**(PODSSymbolScanner* *scanobj*, PODSUInt16 *aimDuration*);
Scan**GetAimDuration**	avantgo.symbolScanner. **AimDuration**	PODSInt16 **PODSgetAimDuration**(PODSSymbolScanner* *scanobj*);
Scan**SetTriggering Modes**	avantgo.symbolScanner. **TriggeringModes**	PODSInt16 **PODSsetTriggeringModes**(PODSSymbolScanner* *scanobj*, PODSUInt16 *triggerMode*);
Scan**GetTriggering Modes**	avantgo.symbolScanner. **TriggeringModes**	PODSInt16 **PODSgetTriggeringModes**(PODSSymbolScanner* *scanobj*);
Scan**SetBeepAfter GoodDecode**	avantgo.symbolScanner. **SetBeepAfterGood Decode**()	PODSInt16 **PODSsetBeepAfterGoodDecode**(PODSSymbolScanner* *scanobj*, PODSBoolean *enable*);
Scan**GetBeepAfter GoodDecode**	avantgo.symbolScanner. **GetBeepAfterGood Decode**()	PODSInt16 **PODSgetBeepAfterGoodDecode**(PODSSymbolScanner* *scanobj*);
Scan **SetLinearCodeType SecurityLevel**	avantgo.symbolScanner. **LinearCodeTypeSecurity Level**	PODSInt16 **PODSsetLinearCodeType SecurityLevel**(PODSSymbolScanner* *scanobj*, PODSUInt16 *securityLevel*);
Scan**GetLinearCode TypeSecurityLevel**	avantgo.symbolScanner. **LinearCodeType SecurityLevel**	PODSInt16 **PODSgetLinearCodeType SecurityLevel**(PODSSymbolScanner* *scanobj*);
Scan**SetBidirectional Redundancy**	avantgo.symbolScanner. **Bidirectional Redundancy**	PODSInt16 **PODSsetBidirectional Redundancy**(PODSSymbolScanner* *scanobj*, PODSUInt16 *redundancy*);

Symbol Technologies method	M-Business JavaScript synopsis	M-Business C synopsis
Scan**GetBidirectional Redundancy**	avantgo.symbolScanner. **Bidirectional Redundancy**	PODSInt16 **PODSgetBidirectional Redundancy**(PODSSymbolScanner* *scanobj*);
Scan**SetTransmitCode IdCharacter**	avantgo.symbolScanner. **TransmitCodeId Character**	PODSInt16 **PODSsetTransmit CodeIdCharacter**(PODSSymbolScanner* *scanobj*, PODSUInt16 *codeId*);
Scan**GetTransmit CodeId Character**	avantgo.symbolScanner. **TransmitCodeId Character**	PODSInt16 **PODSgetTransmitCodeId Character**(PODSSymbolScanner* *scanobj*);
Scan**SetPrefixSuffix Values**	avantgo.symbolScanner. **SetPrefixSuffix**()	PODSInt16 **PODSsetPrefixSuffix**(PODSSymbolScanner* *scanobj*, PODSUInt16 *prefix*, PODSUInt16 *suffix1*, PODSUInt16 *suffix2*);
Scan**GetPrefixSuffix Values**	avantgo.symbolScanner. **GetPrefixSuffixPfx**() avantgo.symbolScanner. **GetPrefixSuffixSfx1**() avantgo.symbolScanner. **GetPrefixSuffixSfx2**()	PODSInt16 **PODSgetPrefixSuffixPrefix**(PODSSymbolScanner* *scanobj*); PODSInt16 **PODSgetPrefixSuffixSuffix1**(PODSSymbolScanner* *scanobj*); PODSInt16 **PODSgetPrefixSuffixSuffix2**(PODSSymbolScanner* *scanobj*);
Scan**SetScanData TransmissionFormat**	avantgo.symbolScanner. **ScanDataTransmission Format**	PODSInt16 **PODSsetScanData TransmissionFormat**(PODSSymbolScanner* *scanobj*, PODSUInt16 *transFormat*);
Scan**GetScanData TransmissionFormat**	avantgo.symbolScanner. **ScanDataTransmission Format**	PODSInt16 **PODSgetScanData TransmissionFormat**(PODSSymbolScanner* *scanobj*);

Symbol Technologies method	M-Business JavaScript synopsis	M-Business C synopsis
Scan**SetHostSerial ResponseTimeOut**	avantgo.symbolScanner. **HostSerialResponse TimeOut**	PODSInt16 **PODSsetHostSerial ResponseTimeout**(PODSSymbolScanner* *scanobj*, PODSUInt16 *timeout*);
Scan**GetHostSerial ResponseTimeOut**	avantgo.symbolScanner. **HostSerialResponse TimeOut**	PODSInt16 **PODSgetHostSerial ResponseTimeout**(PODSSymbolScanner* *scanobj*);
Scan**SetBeepFrequency**	avantgo.symbolScanner. **SetBeepFrequency**()	PODSInt16 **PODSsetBeepFrequency**(PODSSymbolScanner* *scanobj*, PODSSymbolFrequencyType *freqType*, PODSInt16 *freq*);
Scan**GetBeepFrequency**	avantgo.symbolScanner. **GetBeepFrequency**()	PODSInt16 **PODSgetBeepFrequency**(PODSSymbolScanner* *scanobj*, PODSSymbolFrequencyType *freqType*);
Scan**SetBeepDuration**	avantgo.symbolScanner. **SetBeepDuration**()	PODSInt16 **PODSsetBeepDuration**(PODSSymbolScanner* *scanobj*, PODSSymbolDurationType *durationType*, PODSInt16 *duration*);
Scan**GetBeepDuration**	avantgo.symbolScanner. **GetBeepDuration**()	PODSInt16 **PODSgetBeepDuration**(PODSSymbolScanner* *scanobj*, PODSSymbolDurationType *durationType*);
Scan**SetTimeOut BetweenSameSymbol**	avantgo.symbolScanner. **TimeOutBetweenSame Symbol**	PODSInt16 **PODSsetTimeOutBetween SameSymbol**(PODSSymbolScanner* *scanobj*, PODSUInt16 *timeout*);
Scan**GetTimeOut BetweenSameSymbol**	avantgo.symbolScanner. **TimeOutBetweenSame Symbol**	PODSInt16 **PODSgetTimeOutBetween SameSymbol**(PODSSymbolScanner* *scanobj*);

CHAPTER 12

PODS event and exception objects

Contents

PODSEventHandler object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSAvantGo` object's "createObject( )" on page 85 to create object that implements `PODSEventHandler` interface, then `PODSEventMgr` object's "registerEventHandler( )" on page 323
- **Available to:** C only

The `PODSEventHandler` interface provides a method to handle an event.

An event is a normal user- or system-generated, platform-specific condition that your application may be expected to encounter. For information on handling exceptions, or error conditions, see "PODSException object" on page 325.

Summary of PODSEventHandler attributes and methods

Description	Attributes and methods
Handling an event	"handleEvent( )" on page 320

handleEvent( )

Handles a `PODSEvent` object.

Interface

`PODSEventHandler`

IDL definition

PODSBoolean handleEvent(PODSEvent *event*);

JavaScript synopsis

Not applicable

C synopsis

PODSBoolean **PODShandleEvent(**
  PODSEventHandler* *evhandler*,
  PODSEvent* *event*
);

Parameters

- ***evhandler*** The `PODSEventHandler` object.
- ***event*** [in] The `PODSEvent` object to handle.

Returns

PODS_TRUE

If the PODSEvent object was successfully handled.

PODS_FALSE

Otherwise.

Remarks

Event handlers should return PODS_TRUE to indicate the event handler has been handled and does not need any further processing.

PODSEventMgr 's implementation of handleEvent() , inherited from PODSEventHandler , sends handleEvent() to each of the event handlers that have been registered. It returns PODS_TRUE if any event handler returns PODS_TRUE , indicating that one of the event handlers has handled the event and that no further processing of the event should occur. See "PODSEventMgr object" on page 322.

See also

PODSEventMgr object's "handleEvent( )" on page 322

PODSEventMgr object

- **Inherits from:** `PODSEventHandler`
- **Accessed by:** `PODSAvantGo` object's "eventMgr" on page 87
- **Available to:** C only

The `PODSEventMgr` methods register and unregister event handlers.

`PODSEventMgr`'s implementation of `handleEvent( )`, inherited from `PODSEventHandler`, sends `handleEvent( )` to each of the event handlers that have been registered. It returns `PODS_TRUE` if any event handler returns `PODS_TRUE`, indicating that one of the event handlers has handled the event and that no further processing of the event should occur. See "handleEvent( )" on page 320.

Summary of PODSEventMgr attributes and methods

Description	Attributes and methods
Registering/unregistering an event handler	"registerEventHandler( )" on page 323 "unregisterEventHandler( )" on page 324
Handling an event	"handleEvent( )" on page 322

handleEvent( )

Handle a `PODSEvent` object.

Interface

`PODSEventMgr`

IDL definition

PODSBoolean handleEvent(PODSEvent *event*);

JavaScript synopsis

Not applicable

C synopsis

```
PODSBoolean PODShandleEvent(
  PODSEventMgr* evmgr,
  PODSEvent* event
);
```

Parameters

- ***evmgr*** The `PODSEventMgr` object.

- ***event*** [in] The `PODSEvent` object to handle.

Returns

`PODS_TRUE`

If the `PODSEvent` object was successfully handled.

`PODS_FALSE`

Otherwise.

Remarks

Event handlers should return `PODS_TRUE` to indicate the event handler has been handled and does not need any further processing.

The `PODSEventMgr` implementation of `handleEvent( )`, inherited from `PODSEventHandler`, sends `handleEvent( )` to each of the event handlers that have been registered. It returns `PODS_TRUE` as soon as an event handler returns `PODS_TRUE`, indicating that the event was handled and that no further processing of the event should occur. The `PODSEventHandler` implementation only checks the specified event handler.

See also

`PODSEventHandler` object's "handleEvent( )" on page 320

registerEventHandler( )

Registers a `PODSEventHandler` object.

Interface

`PODSEventMgr`

IDL definition

void registerEventHandler(PODSEventHandler *evhandler*);

JavaScript synopsis

Not applicable

C synopsis

void **PODSregisterEventHandler(**
  PODSEventMgr* *evmgr*,
  PODSEventHandler* *evhandler*
);

Parameters

- ***evmgr*** The `PODSEventMgr` object.
- ***evhandler*** [in] The `PODSEventHandler` object to register.

Returns

None

See also

"unregisterEventHandler( )" on page 324

unregisterEventHandler( )

Unregisters a `PODSEventHandler` object.

Interface

`PODSEventMgr`

IDL definition

void unregisterEventHandler(PODSEventHandler *evhandler*);

JavaScript synopsis

Not applicable

C synopsis

void **PODSunregisterEventHandler(**
  PODSEventMgr* *evmgr*,
  PODSEventHandler* *evhandler*
);

Parameters

- ***evmgr*** The `PODSEventMgr` object.
- ***evhandler*** [in] The `PODSEventHandler` object to unregister.

Returns

None

See also

"registerEventHandler( )" on page 323

PODSException object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSExceptionMgr` object's "createStringException( )" on page 327
- **Available to:** C, JavaScript

The `PODSException` interface provides a method to get the message associated with an exception.

An exception is not a normal user- or system-generated, platform-specific condition that your application may be expected to encounter. It is an exceptional condition, or an error condition, that you need to handle if it does occur. For information on handling normal, user- or system-generated events, see "PODSObject object" on page 68.

Summary of PODSException attributes and methods

Description	Attributes and methods
Getting an exception message	"getMessage( )" on page 325
Getting an exception description	"toString( )" on page 326

getMessage( )

Get the exception message.

Interface

`PODSException`

IDL definition

PODSString getMessage( );

JavaScript synopsis

except.**getMessage( )**

C synopsis

PODSString **PODSgetMessage(**PODSException* *except***);**

Parameters

- ***except*** The `PODSException` object.

Returns

The exception message.

Remarks

Use the `PODSExceptionMgr` object's "createStringException( )" on page 327 to create exception messages.

See also

`PODSExceptionMgr` object's "throw( )" on page 328

toString( )

Gets the description for the `PODSException` object.

Interface

`PODSException`

IDL definition

PODSString toString( );

JavaScript synopsis

except.**toString( )**

C synopsis

PODSString **PODStoString(**PODSException* *except*);

Parameters

except

The `PODSException` object.

Returns

The description for the `PODSException` object.

See also

"createStringException( )" on page 327

PODSExceptionMgr object

- **Inherits from:** `PODSObject`
- **Accessed by:** `PODSAvantGo` object's "exceptionMgr" on page 88
- **Available to:** C only

`PODSExceptionMgr` provides JavaScript-compatible exception handling for PODS, including the ability to throw an exception to be caught in M-Business JavaScript engine.

Summary of PODSExceptionMgr attributes and methods

Description	Attributes and methods
Throwing/trying exception	"throw( )" on page 328 "try( )" on page 328
Creating exception message string	"createStringException( )" on page 327

createStringException( )

Creates an exception message.

Interface

`PODSExceptionMgr`

IDL definition

PODSException createStringException(PODSString *msg*);

JavaScript synopsis

Not applicable

C synopsis

PODSException **createStringException(**
  PODSExceptionMgr* *exmgr*,
  PODSException* *except*
);

Parameters

- ***exmgr*** The `PODSExceptionMgr` object.
- ***msg*** [in] The message string for the string exception.

Returns

None

throw( )

Throws a `PODSException` object.

Interface

`PODSExceptionMgr`

IDL definition

void throw(PODSException *except*);

JavaScript synopsis

Not applicable

C synopsis

void **PODSthrow(**
PODSExceptionMgr* *exmgr*,
PODSException* *except*
);

Parameters

- ***exmgr*** The `PODSExceptionMgr` object.
- ***except*** [in] The `PODSException` object to throw.

Returns

None

try( )

Begins an exception handler code block.

Interface

`PODSExceptionMgr`

IDL definition

void try(PODSTryBlock *tryblk*);

JavaScript synopsis

Not applicable

C synopsis

void **PODStry(**
PODSExceptionMgr* *exmgr*,


```
  PODSTryBlock* tryblk
);
```

Parameters

- ***exmgr*** The `PODSExceptionMgr` object.
- ***tryblk*** [in] The `PODSTryBlock` object.

Returns

None

CHAPTER 13

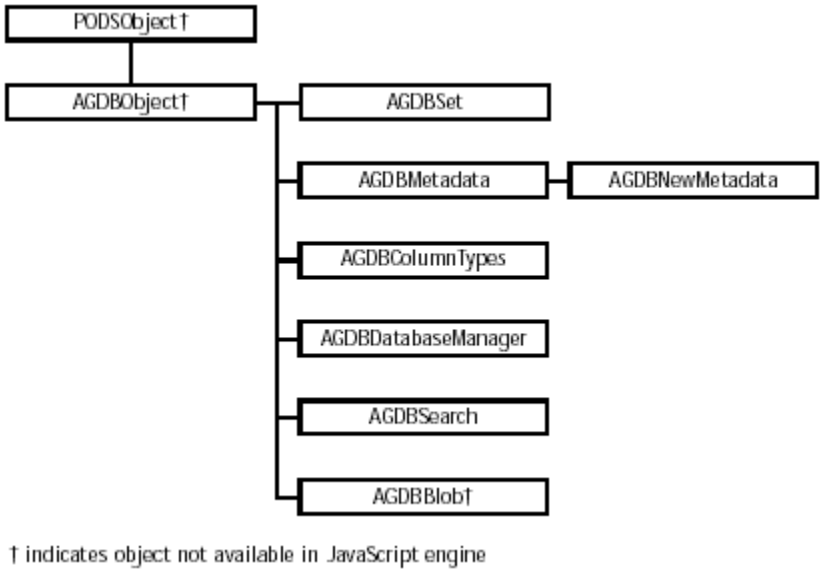
M-Business XML API reference

Contents

Roadmap to M-Business XML API interfaces

Note
In addition to the M-Business XML API, UltraLite for M-Business Anywhere provides a robust alternative that allows two-way data synchronization. For information on the trade-offs between these two options, see "Using UltraLite for M-Business Anywhere for on-device data" [*M-Business Anywhere Application Developer Guide*].

This section summarizes the functionality provided by each M-Business XML API interface.



Available M-Business XML API interfaces

- AGDBObject is the base interface from which all other interfaces extend.
- AGDBSet represents the entire set of records from a table. At any time, the `AGDBSet` object refers to only a single record within the set as the current record. You can create as many `AGDBSet` objects as you need. See "AGDBSet object" on page 335.
- AGDBMetadata describes the columns available in a particular `AGDBSet` object. See "AGDBMetadata object" on page 369.
- AGDBNewMetadata is an extension of the `AGDBMetadata` interface, which allows for adding new columns. See "AGDBNewMetadata object" on page 374.
- AGDBDatabaseManager is the starting point for the `avantgo.db` system. You can use it to open, create, and remove `AGDBSet` objects. See "AGDBDatabaseManager object" on page 382.

- AGDBSearch represents a search to be performed on an `AGDBSet` object. See "AGDBSearch object" on page 388.
- AGDBBlob represents a piece of binary data for storage in an `AGDBSet` object. See "AGDBBlob object" on page 389.

Using the M-Business XML API

Supported data types in XSD files

Refer to the table below when you need to specify data types when working with the M-Business XML API. For information on designing an application and setting up supporting files, see "Using M-Business XML conduit and API" [*M-Business Anywhere Application Developer Guide*] and "Building applications with M-Business XML conduit" [*M-Business Anywhere Application Developer Guide*].

Caution
If you are supporting double-byte character set (DBCS) data, you need to follow some additional guidelines for setting up the supporting files. See "Working with double-byte character sets (DBCS)" [*M-Business Anywhere Application Developer Guide*].

Table 1. Supported data types in XSD files

Data type	XSD type specification
64-bit unsigned integer	`type = "xsd:double"`
32-bit signed integer	`type = "xsd:int"`
32-bit unsigned integer	`type = "xsd:unsignedInt"`
16-bit signed integer	`type = "xsd:short"`
16-bit unsigned integer	`type = "xsd:unsignedShort"`
character string (32kB maximum size)	`type = "xsd:string"`
boolean value (valid values: "`true`", "`false`")	`type = "xsd:boolean"`
date/time value (supports dates from January 1, 1902, to January 1, 2038 — format is: `YYYY:MM:DD:T:hh:mm:ss`)	`type = "xsd:datetime"`

Downloading the supporting code files to user devices

Before your users can synchronize on-device data implemented through the M-Business XML API, a platform-specific database POD must be downloaded to their devices. This download is independent of the channel content that included the data itself. For instructions on downloading the database POD, see "Enabling the download of the M-Business XML POD (database POD) to the group" [*M-Business Anywhere Application Developer Guide*].

AGDBSet object

- **Inherits from:** `AGDBObject`
- **Available to:** JavaScript and C

An `AGDBSet` object represents the entire set of records from a table. At any time, the `AGDBSet` object refers to only a single record within the set as the current record. You can create as many `AGDBSet` objects as you need.

When you open an `AGDBSet` object, the current record is positioned to the first record (if any) and the `atbof( )` and `ateof( )` methods return `FALSE`. If there are no records, the `atbof( )` and `ateof( )` methods return `TRUE`.

Summary of AGDBSet attributes and methods

Description	Attributes and methods
Gathering information about the `AGDBSet` object	"atbof( )" on page 337 "ateof( )" on page 337 "index" on page 350 "metadata" on page 351 "nrows" on page 356
Setting the position of the current record in the `AGDBSet` object	"moveBy( )" on page 352 "moveFirst( )" on page 352 "moveLast( )" on page 353 "moveNext( )" on page 354 "movePrev( )" on page 354 "moveTo( )" on page 355
Making changes to the database	"addNew( )" on page 336 "close( )" on page 338 "deleteRow( )" on page 341 "removeRow( )" on page 356 "rowDeleted( )" on page 357 "rowUpdated( )" on page 358 "undo( )" on page 367

Description	Attributes and methods
Creating a search	"createSearch( )" on page 339 "filterDeleteRecords( )" on page 342 "find( )" on page 343 "getBlobField( )" on page 344 "getDateField( )" on page 345 "setBlobField( )" on page 358 "setBooleanField( )" on page 359 "setDateField( )" on page 360 "setFilter( )" on page 366 "setSort( )" on page 366

addNew( )

Creates a new record and initializes it to zero. The new record becomes the current record and remains current after you call the `commit` method. The new record is appended to the `AGDBSet` object.

Interface

`AGDBSet`

IDL definition

void addNew( );

ADO equivalent

`AddNew` method

JavaScript synopsis

dbset.**addNew( )**

C synopsis

void **AVaddNew(**AGDBSet* *dbset*);

Parameters

- ***dbset*** The `AGDBSet` object.

Returns

None

Remarks

If you call `addNew( )` while editing the current record or while adding a new record, `AGDBSet` interface calls the `undo( )` method to undo any changes and then creates the new record. Note that `addNew( )` cannot be undone; the new record is immediately inserted into the `AGDBSet` object.

See also

"undo( )" on page 367

atbof( )

Returns `TRUE` if the current record position is before the first record in an `AGDBSet` object.

Interface

`AGDBSet`

IDL definition

boolean atbof( );

ADO equivalent

BOF Property

JavaScript synopsis

dbset.**atbof( )**

C synopsis

PODSBoolean **AVatbof(**AGDBSet* *dbset*);

Parameters

- ***dbset*** The `AGDBSet` object.

Returns

`TRUE` if the current record position is before the first record in an `AGDBSet` object; `FALSE` otherwise.

Remarks

If you open an `AGDBSet` object containing no records, the `BOF` property is set to `TRUE`.

ateof( )

Returns `TRUE` if the current record position is after the last record in an `AGDBSet` object.

Interface

`AGDBSet`

IDL definition

boolean ateof( );

ADO equivalent

EOF Property

JavaScript synopsis

dbset.**ateof( )**

C synopsis

PODSBoolean **AVateof(**AGDBSet* *dbset***);**

Parameters

- ***dbset*** The AGDBSet object.

Returns

TRUE if the current record position is after the last record in an AGDBSet object; FALSE otherwise.

Remarks

If you open an AGDBSet object containing no records, the EOF property is set to TRUE.

close( )

Releases the associated data and any exclusive access you may have had to the data through this object.

Interface

AGDBSet

IDL definition

void close( );

ADO equivalent

Close method

JavaScript synopsis

dbset.**close( )**

C synopsis

void **AVclose(**AGDBSet* *dbset***);**

Parameters

- ***dbset*** The AGDBSet object.

Returns

None

Remarks

Subsequent calls to the AGDBSet object after a close( ) will result in an error.

commit( )

Saves any changes made to the current record of an `AGDBSet` object since calling the `addNew( )` method or since changing any field values in an existing record.

Interface

`AGDBSet`

IDL definition

void commit( );

ADO equivalent

Update Method

JavaScript synopsis

dbset.**commit( )**

C synopsis

void **AVcommit(**AGDBSet* *dbset*);

Parameters

- ***dbset*** The `AGDBSet` object.

Returns

None

Remarks

The current record remains current after you call the `commit( )` method.

See also

"addNew( )" on page 336

createSearch( )

Creates an object that can be passed into the `find()` function to locate a record that matches specified criteria. Criteria is a value that contains a statement specifying a single column name, comparison operator, and a value to use in the search.

Interface

`AGDBSet`

IDL definition

AGDBSearch createSearch(String *criteria*);

ADO equivalent

None

JavaScript synopsis

dbset.**createSearch(***criteria***);**

C synopsis

AGDBSearch* **AVcreateSearch(**
AGDBSet* *dbset*,
PODSString *criteria*
);

Parameters

- ***dbset*** The `AGDBSet` object.
- ***criteria*** [in] The search criteria to use. See "Specifying search criteria" on page 340.

Returns

An `AGDBSearch` object that can be passed to `find( )`.

See also

"find( )" on page 343

Specifying search criteria

The comparison operator in *criteria* may be > (greater than), < (less than), = or == (equal), >= (greater than or equal), <= (less than or equal), <> or != (not equal), or `like` or ~ (pattern matching).

You can use `AND` in the specifying search criteria to join multiple filters. For example, you join the two valid filters "`a < b` " and `c < d` with "`a < b and c < d` ". Do not use an expression like "`a AND b` ". It is an invalid expression, since "`a` " and "`b` " are not valid expressions on their own.

The value in *criteria* may be a string, number, date, or `true`. String values may be delimited with single quotes or # (number sign); for example, `"state = 'CA'"` or `"state = #CA#"`. Date values are JavaScript `date.toString( )` formatted; for example, `#Wed Apr 17 14:18:14 GMT+0000 2002# or #Wed Apr 17 14:18:14 GMT-0700 (Pacific Daylight Time) 2002#)`. Dates can also be specified as a number corresponding to the value returned from a JavaScript `Date` object `getTime( )`.

If the comparison operator is `like` or ~ , the string value may contain an asterisk (`*`) to find one or more occurrences of any character or substring. For example, `"state like 'C*'"` matches California and Colorado. You can also use leading and trailing asterisks to find a substring contained within the values. For example, `"state like '*as*'"` matches Alaska, Arkansas, and Massachusetts.

Asterisks can be used only at the end of a criteria string, or together at both the beginning and end of a criteria string, as shown above. You cannot use the asterisk as a leading wildcard (`'*str'`), or embedded wildcard (`'s*r'`). String comparisons are not case sensitive. If you specify `like` and do not include any asterisks, then `*value*` is assumed.

The comparison operator `$=` will do a case-sensitive string search.

The following are all valid JavaScript examples of search criteria:

```
myDate = new Date(2002, 10, 23);
search = myDBSet.createSearch("startDate >= " + myDate);
search = myDBSet.createSearch("startDate >= " + myDate.getTime());
search = myDBSet.createSearch("address like #park lane#");
search = myDBSet.createSearch("name == 'David'");
search = myDBSet.createSearch("city == Chicago");
search = myDBSet.createSearch("zipcode >= 90302");
search = myDBSet.createSearch("hasPaid == true");
search = myDBSet.createSearch("city $= Chicago");
```

deleteRow( )

Immediately marks the current record in the `AGDBSet` object for deletion and sets the data values to zero. The record is physically removed from the `AGDBSet` object during the next synchronization.

Interface

`AGDBSet`

IDL definition

void deleteRow( );

ADO equivalent

`Delete` method

JavaScript synopsis

dbset.**deleteRow( )**

C synopsis

void **AVdeleteRow(**AGDBSet* *dbset*);

Parameters

- ***dbset*** The `AGDBSet` object.

Returns

None

Remarks

After using `deleteRow( )`, the record continues to exist in the `AGDBSet` object and the index does not move in response to this method. Note that `deleteRow( )` cannot be undone.

See also

"removeRow( )" on page 356

filterDeleteRecords( )

Adds a filter to the AGDBSet object that removes any rows marked for deletion.

Interface

AGDBSet

IDL definition

void filterDeleteRecords(boolean *filter*);

ADO equivalent

None

JavaScript synopsis

dbset.**filterDeleteRecords(***filter***)**

C synopsis

void **AVfilterDeletedRecords(**
*dbset***,**
PODSBoolean*filter*
);

Parameters

- ***dbset*** The AGDBSet object.
- ***filter*** Boolean that indicates whether the deleted records should be filtered.

Returns

None

Remarks

This filter will work even with another filter that you may set using setFilter( ). Calling this method will cause the filter loop to run again.

Example

Assume you have a database on device where some records have been marked for deletion. The deletion will not actually be performed until the next synchronization, so you do not want these records to show up in any lists that the user displays. The following code suppresses the display of the records that have been flagged for deletion:

```
...
var DBMgr = CreateObject('avantgo.db');
var DBName = "StressGets";
var DBSet = null;
...
function openDB()
{
if(DBSet)
DBSet.close();
DBSet = gDBMgr.open(DBName, "w");
```



```
DBset.filterDeletedRecords(true);
}
```

See also

"setFilter( )" on page 366

find( )

Within an AGDBSearch object created with createSearch( ), finds the next row that satisfies the search criteria.

Interface

AGDBSet

IDL definition

boolean find(AGDBSearch *searchset*);

ADO equivalent

Find method

JavaScript synopsis

dbset.**find(***searchset***)**

C synopsis

```
PODSBoolean AVfind(
  AGDBSet* dbset,
  AGDBSearch* searchset
);
```

Parameters

- ***dbset*** The AGDBSet object.
- ***searchset*** [in] An AGDBSearch object created with createSearch( ).

Returns

TRUE if a matching record is found; FALSE otherwise.

Remarks

The search criteria are set by createSearch( ). If the criteria are met, the current row position is set on the found record; otherwise, the position is set to the end of the AGDBSet object.

See also

"createSearch( )" on page 339, "moveNext( )" on page 354

Moving the row pointer for the next find( )

Searching starts with the current row and moves forward. Note that you normally move the current row forward yourself after returning from a successful `find( )` to continue finding. The following JavaScript is an example:

```
dbset.find(searchset)
searchset = dbset.createSearch("state == #CA#");
while(dbset.find(searchset)) {
   alert(dbset.state);
   dbset.moveNext();
}
```

getBlobField( )

Gets the `AGDBBlob` object in a field of an `AGDBSet` object.

Interface

`AGDBSet`

IDL definition

AGDBBlob getBlobField(const String *name*);

ADO equivalent

None

JavaScript synopsis

dbset.**getBlobField(***name***)**

C synopsis

AGDBBlob* **AVgetBlobField(**
  AGDBSet* *dbset*,
  const PODSString name
);

Parameters

- ***dbset*** The `AGDBSet` object.
- ***name*** [in] The name of the blob field to get.

Returns

The blob in the named field.

See also

"setBlobField( )" on page 358, "AGDBBlob object" on page 389

getBooleanField( )

Gets the boolean value for a field in an `AGDBSet` object.

Interface

`AGDBSet`

IDL definition

boolean getBooleanField(const String*name*);

ADO equivalent

`Field` object accessors

JavaScript synopsis

dbset.**getBooleanField(***name***)**

C synopsis

```
PODSBoolean AVgetBooleanField(
  AGDBSet* dbset,
  const PODSString name
);
```

Parameters

- ***dbset*** The `AGDBSet` object.
- ***name*** [in] The name of the field.

Returns

Boolean value of field.

See also

"setBooleanField( )" on page 359

getDateField( )

Gets the date value for a field in an `AGDBSet` object.

Interface

`AGDBSet`

IDL definition

PODSDate getDateField(const String *name*);

ADO equivalent

`Field` object accessors

JavaScript synopsis

dbset.**getDateField(***name***)**

C synopsis

PODSDate **AVgetDateField(**
AGDBSet* *dbset*,
const PODSString name
);

Parameters

- ***dbset*** The AGDBSet object.
- ***name*** [in] The name of the field.

Returns

The date value for the field.

Remarks

Calling `getDateField( )` with a column name for a string column will result in an error.

See also

"setDateField( )" on page 360

getDoubleField( )

Gets the doubles value for a field in an AGDBSet object.

Interface

AGDBSet

IDL definition

double getDoubleField(const String *name*);

ADO equivalent

Fields collection on RecordSet

JavaScript synopsis

gDBSet.**getDoubleField(***"doubleField"***);**

C synopsis

PODSDouble **getDoubleField(***const String name***);**

Parameters

- ***gDbset*** The AGDBSet object.
- ***name*** [in] The name of the field.

Returns

The doubles value for the field.

See also

"setDoubleField( )" on page 361

getInt32Field( )

Gets the 32- bit integer value for a field in the `AGDBSet` object.

Interface

`AGDBSet`

IDL definition

long getInt32Field(const String *name*);

ADO equivalent

`Field` object accessors

JavaScript synopsis

dbset.**getInt32Field(***name***)**

C synopsis

```
PODSInt32 AVgetInt32Field(
  AGDBSet* dbset,
  const PODSString name
);
```

Parameters

- ***dbset*** The `AGDBSet` object.
- ***name*** [in] The name of the field.

Returns

The 32-bit integer value for a field.

See also

"setInt32Field( )" on page 361

getUInt32Field( )

Gets the unsigned 32-bit integer value of a field in the `AGDBSet` object.

Interface

`AGDBSet`

IDL definition

unsigned long getUInt32Field(const String *name*);

ADO equivalent

`Field` object accessors

JavaScript synopsis

dbset.**getUInt32Field(***name***)**

C synopsis

PODSInt32 **AVgetUInt32Field(**
  AGDBSet* *dbset*,
  const PODSString *name*
);

Parameters

- ***dbset*** The `AGDBSet` object.
- ***name*** [in] The name of the field.

Returns

The value of the field as an unsigned 32-bit integer value.

See also

"setUInt32Field( )" on page 362

getInt16Field( )

Gets the 16-bit integer value of a field in the `AGDBSet` object.

Interface

`AGDBSet`

IDL definition

short getInt16Field(const String *name*);

ADO equivalent

`Field` object accessors

JavaScript synopsis

dbset.**getInt16Field(***name***)**

C synopsis

PODSInt16 **AVgetInt16Field(**
  AGDBSet* *dbset*,
  const PODSString *name*
);

Parameters

- ***dbset*** The `AGDBSet` object.
- ***name*** [in] The name of the field.

Returns

The value of the field as a 16-bit integer value.

See also

"setInt16Field( )" on page 363

getUInt16Field( )

Gets the value of the field as an unsigned 16-bit integer value.

Interface

`AGDBSet`

IDL definition

unsigned short getUInt16Field(const String *name*);

ADO equivalent

`Field` object accessors

JavaScript synopsis

dbset.**getUInt16Field(***name***)**

C synopsis

```
PODSUInt16 AVgetUInt16Field(
  AGDBSet* dbset,
  const PODSString name
);
```

Parameters

- ***dbset*** The `AGDBSet` object.
- ***name*** [in] The name of the field.

Returns

The value of the field as an unsigned 16-bit integer value.

See also

"setUInt16Field( )" on page 364

getStringField( )

Gets the string value of a field in the `AGDBSet` object.

Interface

`AGDBSet`

IDL definition

`String getStringField(const String` *name*`);`

ADO equivalent

`Field` object accessors

JavaScript synopsis

dbset.**getStringField(***name***)**

C synopsis

PODSString **AVgetStringField(**
  AGDBSet* *dbset*,
  const PODSString *name*
);

Parameters

- ***dbset*** The `AGDBSet` object.
- ***name*** [in] The name of the field.

Returns

The string value of a field.

See also

"setStringField( )" on page 365

index

Indicates the ordinal position of an `AGDBSet` object's current record.

Interface

`AGDBSet`

IDL definition

readonly attribute unsigned long index;

ADO equivalent

`AbsolutePosition` method

JavaScript synopsis

dbset.**index**

C synopsis

PODSUInt32 **AVgetIndex(**AGDBSet* *dbset***);**

Parameters

- ***dbset*** The `AGDBSet` object.

Returns

A value from `1` to `nrows` in the `AGDBSet` object.

See also

"nrows" on page 356

metadata

The metadata describing the `AGDBSet` object.

Interface

`AGDBSet`

IDL definition

readonly attribute AGDBMetadata metadata;

ADO equivalent

Fields property

JavaScript synopsis

dbset.**metadata**

C synopsis

AGDBMetadata* **AVgetMetadata(**AGDBSet* *dbset***);**

Parameters

- ***dbset*** The `AGDBSet` object.

Returns

The metadata describing this `AGDBSet` object.

See also

"createMetadata( )" on page 383, "appendColumn( )" on page 374

moveBy( )

Moves the position of the current record in an AGDBSet object by a specified number of rows.

Interface

AGDBSet

IDL definition

boolean moveBy(long *numRecords*);

ADO equivalent

Move method

JavaScript synopsis

dbset.**moveBy(***numRecords***)**

C synopsis

```
PODSBoolean AVmoveBy(
  AGDBSet* dbset,
  PODSInt32 numRecords
);
```

Parameters

- ***dbset*** The AGDBSet object.
- ***numRecords*** [in] The number of rows to move by.

Returns

TRUE on success; FALSE otherwise.

Remarks

If the *numRecords* argument is greater than zero, the current record position moves forward (toward the end of the AGDBSet object). If the *numRecords* is less than zero, the current record position moves backward (toward the beginning of the AGDBSet object).

moveFirst( )

Moves the current record position to the first record in the AGDBSet object.

Interface

AGDBSet

IDL definition

boolean moveFirst( );

ADO equivalent

`moveFirst` method

JavaScript synopsis

dbset.**moveFirst( )**

C synopsis

PODSBoolean **AVmoveFirst(**AGDBSet* *dbset*);

Parameters

- ***dbset*** The `AGDBSet` object.

Returns

`TRUE` on success; `FALSE` otherwise.

See also

"atbof( )" on page 337

moveLast( )

Moves the current record position to the last record in the `AGDBSet` object.

Interface

`AGDBSet`

IDL definition

boolean moveLast( );

ADO equivalent

`moveLast` method

JavaScript synopsis

dbset.**moveLast( )**

C synopsis

PODSBoolean **AVmoveLast(**AGDBSet* *dbset*);

Parameters

- ***dbset*** The `AGDBSet` object.

Returns

`TRUE` on success; `FALSE` otherwise.

See also

"ateof( )" on page 337

moveNext( )

Moves the current record position one record forward (toward the bottom of the AGDBSet object).

Interface

AGDBSet

IDL definition

boolean moveNext( );

ADO equivalent

moveNext method

JavaScript synopsis

dbset.**moveNext( )**

C synopsis

PODSBoolean **AVmoveNext(**AGDBSet* *dbset***);**

Parameters

- ***dbset*** The AGDBSet object.

Returns

TRUE on success; FALSE otherwise.

Remarks

If the last record is the current record and you call the moveNext method, the AGDBSet interface sets the current record to the position after the last record in the AGDBSet object, where ateof( ) is TRUE.

See also

"ateof( )" on page 337, "movePrev( )" on page 354

movePrev( )

Moves the current record position one record backward (toward the top of the AGDBSet object).

Interface

AGDBSet

IDL definition

boolean movePrev( );

ADO equivalent

movePrevious

JavaScript synopsis

dbset.**movePrev( )**

C synopsis

PODSBoolean **AVmovePrev(**AGDBSet* *dbset*);

Parameters

- ***dbset*** The AGDBSet object.

Returns

TRUE on success; FALSE otherwise.

Remarks

If the first record is the current record and you call the movePrev method, the AGDBSet interface sets the current record to the position before the first record in the AGDBSet object, where atbof( ) is TRUE.

See also

"atbof( )" on page 337, "moveNext( )" on page 354

moveTo( )

Moves the position of the current record in an AGDBSet object.

Interface

AGDBSet

IDL definition

boolean moveTo(unsigned long *index*);

JavaScript synopsis

dbset.**moveTo(***index***)**

C synopsis

```
PODSBoolean AVmoveTo(
  AGDBSet* dbset,
  PODSUInt32 index
);
```

Parameters

- ***dbset*** The AGDBSet object.
- ***index*** [in] Index of the row to move to.

Returns

TRUE on success; FALSE otherwise.

Remarks

Moves the position of the current record in an `AGDBSet` object. In ADO, this is accomplished by setting the `AbsolutePosition` property to the index you want to move to.

See also

"index" on page 350

nrows

Returns the total number of rows in the `AGDBSet` object.

Interface

AGDBSet

IDL definition

readonly attribute unsigned long nrows;

ADO equivalent

`RecordCount`

JavaScript synopsis

dbset.**nrows**

C synopsis

PODSUInt32 **AVgetNrows(**AGDBSet* *dbset***);**

Parameters

- ***dbset*** The `AGDBSet` object.

Returns

Total number of rows in the `AGDBSet` object.

See also

"index" on page 350

removeRow( )

Removes the record from the `AGDBSet` object. The server will not be informed of this removal and may expect this record to be there during subsequent synchronizations.

Interface

`AGDBSet`

IDL definition

void removeRow( );

JavaScript synopsis

dbset.**removeRow( )**

C synopsis

void **AVremoveRow(**AGDBSet* *dbset***);**

Parameters

- ***dbset*** The AGDBSet object.

Returns

None

Remarks

This method is provided mainly for those databases that only exist on the device, where reporting changes to the server is unwanted or unnecessary.

See also

"deleteRow( )" on page 341

rowDeleted( )

Returns TRUE if the current record is marked for deletion on the next synchronization.

Interface

AGDBSet

IDL definition

boolean rowDeleted( );

ADO equivalent

Status Property == adRecDeleted

JavaScript synopsis

dbset.**rowDeleted**

C synopsis

PODSBoolean **AVrowDeleted(**AGDBSet* *dbset***);**

Parameters

- ***dbset*** The AGDBSet object.

Returns

TRUE if the current record is marked for deletion on the next synchronization; FALSE otherwise.

See also

"deleteRow( )" on page 341, "rowUpdated( )" on page 358

rowUpdated( )

Returns TRUE if the current record has been modified since the last synchronization.

Interface

AGDBSet

IDL definition

boolean rowUpdated( );

ADO equivalent

Status Property == adRecModified

JavaScript synopsis

dbset.**rowUpdated**

C synopsis

PODSBoolean **AVrowUpdated(**AGDBSet* *dbset*);

Parameters

- ***dbset*** The AGDBSet object.

Returns

TRUE if the current record has been modified since the last synchronization; FALSE otherwise.

See also

"rowDeleted( )" on page 357

setBlobField( )

Sets the AGDBBlob data in a field of an AGDBDet object.

Interface

AGDBBlob

IDL definition

```
nometadata void setBlobField(const String name,
  const data ,
  long length
);
```


ADO equivalent

None

JavaScript synopsis

Not applicable

C synopsis

```
void AVsetBlobField(
  AGDBSet* dset,
  const PODSString name,
  const AGDBByte* data,
  PODSInt32 length
);
```

Parameters

- ***dset*** The `AGDBSet` object.
- ***name*** [in] The field name.
- ***data*** [in] The blob data.
- ***length*** [in] The length of the blob data.

Returns

None

See also

"getBlobField( )" on page 344

setBooleanField( )

Sets the boolean value for a field in the `AGDBSet` object.

Interface

`AGDBSet`

IDL definition

```
void setBooleanField(
  const String name,
  boolean value
);
```

ADO equivalent

`Field` object accessors

JavaScript synopsis

dset.**setBooleanField(***name*, *value***)**

C synopsis

```
void AVsetBooleanField(
  AGDBSet* dbset,
  PODSString name,
  PODSBoolean value
);
```

Parameters

- ***dbset*** The `AGDBSet` object.
- ***name*** [in] The name of the field.
- ***value*** [in] The boolean value to set for the field.

Returns

None

See also

"getBooleanField( )" on page 345

setDateField( )

Sets the date value for a field in the `AGDBSet` object.

Interface

`AGDBSet`

IDL definition

```
void setDateField(
  const String name,
  PODSDate value
);
```

ADO equivalent

`Field` object accessors

JavaScript synopsis

dbset.**setDateField(***name*, *value***)**

C synopsis

```
void AVsetDateField(
  AGDBSet* dbset,
  const PODSString name,
  PODSDate value
);
```

Parameters

- ***dbset*** The `AGDBSet` object.

- ***name*** [in] The name of the field.
- ***value*** [in] The date value to set for the field.

Returns

None

See also

"getDateField( )" on page 345

setDoubleField( )

Sets the doubles value for a field in an `AGDBSet` object.

Interface

`AGDBSet`

IDL definition

double setDoubleField(const String *name*);

ADO equivalent

Fields collection on `RecordSet`

JavaScript synopsis

gDBSet.**setDoubleField(**"*doubleField*"**);**

C synopsis

PODSDouble **setDoubleField(***const String name***);**

Parameters

- ***gDbset*** The `AGDBSet` object.
- ***name*** [in] The name of the field.

Returns

The doubles value for the field.

See also

"getDoubleField( )" on page 346

setInt32Field( )

Sets the value, as a 32-bit integer, for a field in the `AGDBSet` object.

Interface

AGDBSet

IDL definition

```
void setInt32Field(
  const String name,
  long value
);
```

ADO equivalent

Field object accessors

JavaScript synopsis

dbset.**setInt32Field(***name***,** *value***)**

C synopsis

```
void AVsetInt32Field(
  AGDBSet* dbset,
  const PODSString name,
  PODSInt32 value
);
```

Parameters

- ***dbset*** The AGDBSet object.
- ***name*** [in] The name of the field.
- ***value*** [in] The value to set for the field.

Returns

None

See also

"getInt32Field( )" on page 347

setUInt32Field( )

Sets the value, as an unsigned 32-bit integer, for a field in the AGDBSet object.

Interface

AGDBSet

IDL definition

```
void setUInt32Field(
  const String name,
  unsigned long value
);
```


ADO equivalent

`Field` object accessors

JavaScript synopsis

dbset.**setUInt32Field(***name***,** *value***)**

C synopsis

void **AVsetUInt32Field(**
AGDBSet* *dbset***,**
const PODSString *name***,**
PODSUInt32 *value*
);

Parameters

- ***dbset*** The `AGDBSet` object.
- ***name*** [in] The name of the field.
- ***value*** [in] The value to set for the field.

Returns

None

See also

"getUInt32Field( )" on page 347

setInt16Field( )

Sets the value, as a 16-bit integer, for a field in the `AGDBSet` object.

Interface

`AGDBSet`

IDL definition

void setInt16Field(
const String *name*,
short *value*
);

ADO equivalent

`Field` object accessors

JavaScript synopsis

dbset.**setIntField(***name***,** *value***)**

C synopsis

void **AVsetInt16Field(**
AGDBSet* *dbset***,**

```
  const PODSString name,
  PODSInt16 value
);
```

Parameters

- ***dbset*** The AGDBSet object.
- ***name*** [in] The name of the field.
- ***value*** [in] The value to set for the field.

Returns

None

See also

"getInt16Field( )" on page 348

setUInt16Field( )

Sets the value, as an unsigned 16-bit integer, for a field in the AGDBSet object.

Interface

AGDBSet

IDL definition

```
void setUInt16Field(
  const String name,
  unsigned short value
);
```

ADO equivalent

Field object accessors

JavaScript synopsis

dbset.**setUInt16Field(***name***,** *value***)**

C synopsis

```
void AVsetUInt16Field(
  AGDBSet* dbset,
  const PODSString name,
  PODSUInt16 value
);
```

Parameters

- ***dbset*** The AGDBSet object.
- ***name*** [in] The name of the field.
- ***value*** [in] The value to set for the field.

Returns

None

See also

"getUInt16Field( )" on page 349

setStringField( )

Sets the string value for a field in the `AGDBSet` object.

Interface

`AGDBSet`

IDL definition

```
void setStringField(
  const String name,
  const String value
);
```

ADO equivalent

`Field` object accessors

JavaScript synopsis

dbset.**setStringField(***name***,** *value***)**

C synopsis

```
void AVsetStringField(
  AGDBSet* dbset,
  const PODSString name,
  const PODSString value
);
```

Parameters

- ***dbset*** The `AGDBSet` object.
- ***name*** [in] The name of the field.
- ***value*** [in] The string value to set for the field.

Returns

None

See also

"getStringField( )" on page 350

setFilter( )

Takes an AGDBSearch object built from createSearch( ) and creates an internal list of all the records that return TRUE.

Interface

AGDBSet

IDL definition

void setFilter(*searchObj*);

ADO equivalent

None

JavaScript synopsis

dbset.**setFilter(***searchObj***)**

C synopsis

void **AVsetFilter(**
  AGDBSet* *dbset*,
  AGDBSearch* *searchObj*
);

Parameters

- ***dbset*** The AGDBSet object.
- ***searchObj*** [in] Null, or an AGDBSearch object created with the createSearch( ) method.

Returns

A list of records that return TRUE.

Remarks

You can call addNew( ) and deleteRow( ) while a filter is active. These rows will show up in the filtered dbset regardless of their data values. Call setFilter( ) to make sure these new/deleted rows are matching your filter. Calling setFilter(null) removes the current filter.

See also

"createSearch( )" on page 339, "addNew( )" on page 336, "deleteRow( )" on page 341

setSort( )

Creates an internal list of the records in the AGDBSet object and sorts them according to the specified column and order.

Interface

AGDBSet

IDL definition

```
void setSort(
  const String name,
  boolean ascendSort
);
```

ADO equivalent

None

JavaScript synopsis

dbset.**setSort(***name*, *ascendSort***)**

C synopsis

```
void AVsetSort(
  AGDBSet* dbset,
  const PODSString name,
  PODSBoolean ascendSort
);
```

Parameters

- ***dbset*** The `AGDBSet` object.
- ***name*** [in] The name of the field by which to sort.
- ***ascendSort*** [in] Boolean indicating whether to sort in ascending or descending order.

Returns

None

Remarks

You can call `addNew( )` and `deleteRow( )` while a sort is active. These rows will show up at the end of the `AGDBSet` object regardless of their data values. Call `setSort( )` to make sure these new/deleted rows are in the correct order. If a filter is active, only those records will be sorted. Only one sort is active at a time. If you call `setSort( )` twice, the original sort is removed and the second one becomes active.

Note: For performance reasons, the size of the string values to be sorted is limited to the first 1024 characters for Palm OS and 1024*10 for WinCE.

See also

"addNew( )" on page 336, "deleteRow( )" on page 341, "createSearch( )" on page 339

undo( )

Cancels the changes made to the current row.

Interface

`AGDBSet`

IDL definition

void undo( );

ADO equivalent

`CancelUpdate` method

JavaScript synopsis

dbset.**undo( )**

C synopsis

void **AVundo(**AGDBSet* *dbset***);**

Parameters

- ***dbset*** The `AGDBSet` object.

Returns

None

Remarks

Note that you cannot cancel changes to the current row after you call the `commit( )` method.

See also

"commit( )" on page 339

AGDBMetadata object

- **Inherits from:** AGDBSet
- **Accessed by:** AGBDSet.metadata property
- **Available to:** JavaScript and C

The AGDBMetadata object describes the columns available in a particular AGDBSet object. You can set flags for each column in the metadata and for each cell in the actual data.

Adding NULL values

Databases can now contain flagged columns, non-flagged columns, or both. This enables you to add NULL values to the database by setting metadata <mf> flags for each column and data flags <df> for each cell in the actual data. The database containing columns with NULL values have extra data, so the metadata and the actual data is now different. For example, see the following.

Metadata

```
u'<mf><fixed_1>s'<mf><string_2>u'<mf><fixed_3>
    s'<mf><string_4>u'<mf><fixed_5>
    Sn'<mf><string width><string_n>\0
```

Where x' = x with the high bit (0x80) set. <mf> is the metadata flag and is only present if the high bit is set on x.

Data

```
<df>fixed_1 <df><uint16 len>string_2\0 <df>fixed_3
    <df><uint16 len>string_4\0 <df>fixed_5
    <df>string_n\0<NULL PADDING>
```

Where <df> is the data flag and only exists if <mf> exists.

Summary of AGDBMetadata attributes and methods

Description	Attributes and methods
Accessing information about the columns in the AGDBSet object	“getColumnIndex( )” on page 370 “getColumnName( )” on page 370 “getColumnSize( )” on page 371 “getColumnType( )” on page 372 “ncolumns” on page 372

getColumnIndex( )

Returns the index value for a column with a specified name.

Interface

`AGDBMetadata`

IDL definition

const String getColumnIndex(const string *name*);

ADO equivalent

None

JavaScript synopsis

mdObj.**getColumnIndex(***name***)**

C synopsis

```
PODSInt32 AVgetColumnIndex(
  AGDBMetadata* mdObj,
  const PODSString name
);
```

Parameters

- ***mdObj*** The AGDBMetadata object.
- ***name*** [in] The name of the column.

Returns

The index value for the named column.

getColumnName( )

Retrieves the name of the column at the specified index value.

Interface

`AGDBMetadata`

IDL definition

const String getColumnName(const unsigned long *index*);

ADO equivalent

`Name` property on `Field` object

JavaScript synopsis

mdObj.**getColumnName(***index***)**

C synopsis

```
PODSString AVgetColumnName(
  AGDBMetadata* mdObj,
  const PODSUInt32 index
);
```

Parameters

- ***mdObj*** The AGDBMetadata object.
- ***index*** [in] Index of a column.

Returns

The name of the column at the index value.

getColumnSize( )

Retrieves the size of a column for a specified index value.

Interface

`AGDBMetadata`

IDL definition

const String getColumnSize(const unsigned long *index*);

ADO equivalent

`DefinedSize` property on `Field` object

JavaScript synopsis

mdObj.**getColumnSize(***index***)**

C synopsis

```
PODSInt32 AVgetColumnSize(
  AGDBMetadata* mdObj,
  const PODSUInt32 index
);
```

Parameters

- ***mdObj*** The AGDBMetadata object.
- ***index*** [in] The index value of the column.

Returns

The column size, in bytes.

Remarks

String values that have no defined size will return -1.

getColumnType( )

Retrieves the type of the column at a specified index value.

Interface

`AGDBMetadata`

IDL definition

const short getColumnType(const unsigned long *index*);

ADO equivalent

`Type` property on `Field` object

JavaScript synopsis

mdObj.**getColumnType(***index***)**

C synopsis

PODSInt16 **AVgetColumnType(**
  AGDBMetadata* *mdObj***,**
  PODSUInt32 *index*
);

Parameters

- ***mdObj*** The AGDBMetadata object.
- ***index*** [in] The index value of the column.

Returns

Values from the `AGDBColumnTypes` object are available from `AGDBDatabaseManager.types`.

ncolumns

Returns the number of columns defined in the `metadata`.

Interface

`AGDBMetadata`

IDL definition

readonly attribute unsigned long ncolumns;

ADO equivalent

Count

JavaScript synopsis

mdObj.**ncolumns**

C synopsis

PODSUInt32 **AVgetNcolumns(**AGDBMetadata* *mdObj***);**

Parameters

- ***mdObj*** The AGDBMetadata object.

Returns

The number of columns as defined in the `metadata`.

See also

"metadata" on page 351

AGDBNewMetadata object

- **Inherits from:** AGDBMetadata
- **Available to:** C only

The AGDBNewMetadata object is an extension of the AGDBMetadata object that allows for appending a column. The AGDBMetadata object is immutable.

Adding NULL values

Databases can now contain flagged columns, non-flagged columns, or both. This enables you to add NULL values to the database by setting metadata <mf> flags for each column and data flags <df> for each cell in the actual data. The database containing columns with NULL values have extra data, so the metadata and the actual data are now different. For example, see the following.

Metadata

```
u'<mf><fixed_1>s'<mf><string_2>u'<mf><fixed_3>s'
    <mf><string_4>u'<mf><fixed_5>Sn'
    <mf><string width><string_n>\0
```

Where x' = x with the high bit (0x80) set and <mf> is the metadata flag and is only present if the high bit is set on x.

Data

```
<df>fixed_1 <df><uint16 len>string_2\0
    <df>fixed_3 <df><uint16 len>string_4\0
    <df>fixed_5 <df>string_n\0<NULL PADDING>
```

Where <df> is the data flag and only exists if <mf> exists.

Summary of AGDBNewMetadata attributes and methods

Description	Attributes and methods
Adding a column to the AGDB-Set object	“appendColumn( )” on page 374

appendColumn( )

Appends a new column to the end of the metadata.

Interface

AGDBMetadata

IDL definition

```
void appendColumn(
  const String name,
  const short type,
  const unsigned long size
);
```

ADO equivalent

`appendColumn` on `Fields` collection

JavaScript synopsis

mdNewObj.**appendColumn(***name*, *type*,*size***)**

C synopsis

```
void AVappendColumn(
  AGDBNewMetadata* mdNewObj,
  PODSString name,
  PODSInt16 type,
  PODSUInt32 size
);
```

Parameters

- ***mdNewObj*** The AGDBNewMetadata object.
- ***name*** [in] The name of a column.
- ***type*** [in] The type of column.
- ***size*** [in] The size of the column.

Returns

None

Remarks

If creating a column of type `STRINGN`, `size` should be non-zero and less than 255.

See also

"AGDBColumnTypes object" on page 376, "AGDBMetadata object" on page 369

AGDBColumnTypes object

- **Inherits from:** `AGDBObject`
- **Available to:** JavaScript and C

The `AGDBColumnTypes` object contains values for column types. The ADO equivalent is `DataTypeEnum`.

Summary of AGDBColumnTypes attributes and methods

Description	Attributes and methods
Getting constant values corresponding to the column types	"BOOLEAN" on page 376 "DATE" on page 377 "DOUBLE" on page 377 "INT16" on page 378 "UINT16" on page 380 "INT32" on page 378 "UINT32" on page 381 "STRING" on page 379 "STRINGN" on page 380

BOOLEAN

Gets the constant corresponding to the boolean column type.

Interface

`AGDBColumnTypes`

IDL definition

readonly attribute short BOOLEAN;

ADO equivalent

None

JavaScript synopsis

typesObj.**BOOLEAN**

C synopsis

PODSInt16 **AVgetBOOLEAN(**AGDBColumnTypes* *typesObj*)**;**

Parameters

- ***typesObj*** The AGDBColumnTypes object.

Returns

The constant that corresponds to the boolean column type.

DATE

Gets the constant corresponding to the date column type.

Interface

`AGDBColumnTypes`

IDL definition

readonly attribute DATE;

ADO equivalent

None

JavaScript synopsis

typesObj.**DATE**

C synopsis

PODSInt16 **AVgetDATE(**AGDBColumnTypes* *typesObj***);**

Parameters

- ***typesObj*** The AGDBColumnTypes object.

Returns

The constant corresponding to the date column type.

DOUBLE

Gets the constant corresponding to the boolean column type.

Interface

`AGDBColumnTypes`

IDL definition

readonly attribute DOUBLE;

ADO equivalent

None

JavaScript synopsis

typesObj.**DOUBLE**

C synopsis

PODSInt16 **AVgetDOUBLE(**AGDBColumnTypes* *typesObj***);**

Parameters

♦ ***typesObj*** The AGDBColumnTypes object.

Returns

The constant corresponding to the double column type.

INT16

Gets the constant corresponding to the `INT16` column type.

Interface

`AGDBColumnTypes`

IDL definition

readonly attribute INT16;

ADO equivalent

None

JavaScript synopsis

typesObj.**UINT16**

C synopsis

PODSInt16 **AVgetINT16(**AGDBColumnTypes* *typesObj***);**

Parameters

♦ ***typesObj*** The AGDBColumnTypes object.

Returns

The constant that corresponds to the `INT16` type.

INT32

Gets the constant that corresponds to the `INT32` column type.

Interface

`AGDBColumnTypes`

IDL definition

readonly attribute INT32;

ADO equivalent

None

JavaScript synopsis

typesObj.**INT32**

C synopsis

PODSInt16 **AVgetINT32(**AGDBColumnTypes* *typesObj***);**

Parameters

- ***typesObj*** The AGDBColumnTypes object.

Returns

The constant that corresponds to the `INT32` column type.

STRING

Gets the constant that corresponds to the `STRING` column type.

Interface

`AGDBColumnTypes`

IDL definition

readonly attribute STRING;

ADO equivalent

None

JavaScript synopsis

typesObj.**STRING**

C synopsis

PODSInt16 **AVgetSTRING(**AGDBColumnTypes* *typesObj***);**

Parameters

- ***typesObj*** The AGDBColumnTypes object.

Returns

The constant that corresponds to the `STRING` column type.

STRINGN

Gets the constant that corresponds to the `STRINGN` column type.

Interface

`AGDBColumnTypes`

IDL definition

readonly attribute STRINGN;

ADO equivalent

None

JavaScript synopsis

typesObj.**STRINGN**

C synopsis

PODSInt16 **AVgetSTRINGN(**AGDBColumnTypes* *typesObj***);**

Parameters

- ***typesObj*** The AGDBColumnTypes object.

Returns

The constant that corresponds to the `STRINGN` column type.

UINT16

Gets the constant that corresponds to the `UINT16` column type.

Interface

`AGDBColumnTypes`

IDL definition

readonly attribute UINT16;

ADO equivalent

None

JavaScript synopsis

typesObj.**UINT16**

C synopsis

PODSInt16 **AVgetUINT16(**AGDBColumnTypes* *typesObj***);**

Parameters

- ***typesObj*** The AGDBColumnTypes object.

Returns

The constant that corresponds to the `UINT16` column type.

UINT32

Gets the constant that corresponds to the `UINT32` column type.

Interface

`AGDBColumnTypes`

IDL definition

readonly attribute UINT32;

ADO equivalent

None

JavaScript synopsis

typesObj.**UINT32**

C synopsis

PODSInt16 **AVgetUINT32(**AGDBColumnTypes* *typesObj***);**

Parameters

- ***typesObj*** The AGDBColumnTypes object.

Returns

The constant that corresponds to the `UINT32` column type.

AGDBDatabaseManager object

- **Inherits from:** `AGDBObject`
- **Available to:** JavaScript and C

The `AGDBDatabaseManager` object is the starting point for the `avantgo.db` system. It is the object that you get back when you ask for the `avantgo.db` object in JavaScript. The `AGDBDatabaseManager` interface allows you to manage the set of `AGDBSet` objects on the device.

Summary of AGDBDatabaseManager attributes and methods

Description	Attributes and methods
Managing the set of `AGDBSet` objects	“create( )” on page 382 “createMetadata( )” on page 383 “exists( )” on page 384 “open( )” on page 385 “remove( )” on page 386 “types” on page 387

create( )

Creates a new empty `AGDBSet` object on the device.

Interface

`AGDBDatabaseManager`

IDL definition

```
boolean create(
  const String dbname,
  AGDBMetadata mdObj,
  const String flags
);
```

ADO equivalent

None

JavaScript synopsis

dbmgr.**create(***dbname*, *mdObj*, *flags***)**

C synopsis

```
PODSBoolean AVcreate(
  AGDBDatabaseManager* dbmgr,
  const PODSString dbname,
  AGDBMetadata* mdObj,
  const PODSString flags
);
```

Parameters

- ***dbmgr*** The `AGDBDatabaseManager` object.
- ***dbname*** [in] The database name.

  Note: Limited to 30 characters.
- **flags** [in] Flags passed:

  `r` - read mode

  `w` - write mode

  `x` - exclusive mode

  `s` - showSecret mode

  Note: Currently ignored under WinCE and Palm OS.
- ***mdObj*** The `AGDBMetadata` object.

Returns

`TRUE` if successful; `FALSE` otherwise.

createMetadata( )

Creates a `AGDBNewMetadata` suitable for passing into `create( )`.

Interface

`AGDBDatabaseManager`

IDL definition

```
boolean create(
  const String dbname,
  AGDBMetadata mdObj,
  const String flags
);
```

ADO equivalent

None

JavaScript synopsis

dbmgr.**create(***dbname*, *mdObj*, *flags***)**

C synopsis

```
AGDBNewMetadata* AVcreateMetadata(
  AGDBDatabaseManager* dbmgr,
  const PODSString dbname,
  AGDBMetadata* mdObj,
  const PODSString flags
);
```

Parameters

- ***dbmgr*** The `AGDBDatabaseManager` object.
- ***dbname*** [in] Limited to 30 characters.
- ***mdObj*** [in] The `AGDBDatabaseManager` object (description of the columns included in this database).
- ***flags*** [in] Flags passed:

  `r` - read mode

  `w` - write mode

  `x` - exclusive mode

  `s` - showSecret mode

  Note: Currently ignored under WinCE.

Returns

`TRUE` if successful; `FALSE` otherwise.

See also

"AGDBNewMetadata object" on page 374

exists( )

Returns `TRUE` if `dbname` exists.

Interface

`AGDBDatabaseManager`

IDL definition

boolean exists(const String *dbname*);

ADO equivalent

None

JavaScript synopsis

dbmgr.**exists(***dbname***)**

C synopsis

```
PODSBoolean AVexists(
  AGDBDatabaseManager* dbmgr,
  const PODSString dbname
);
```

Parameters

- ***dbmgr*** The `AGDBDatabaseManager` object.
- ***dbname*** [in] The name of the database to test for.

Returns

`TRUE` if named database exists; `FALSE` otherwise.

open( )

Opens the named `AGDBSet` object.

Interface

`AGDBDatabaseManager`

IDL definition

```
AGDBSet open(
  const String dbname,
  const String flags
);
```

ADO equivalent

`Open` method on `RecordSet`

JavaScript synopsis

dbmgr.**open(***dbname*, *flags***)**

C synopsis

```
AGDBSet* AVopen(
  AGDBDatabaseManager* dbmgr,
  const PODSString dbname,
  const PODSString flags
);
```

Parameters

- ***dbmgr*** The `AGDBDatabaseManager` object.
- ***dbname*** [in] The name of the database.
- ***flags*** [in] Flags passed:

  `r` - read mode

`w` - write mode

`x` - exclusive mode

`s` - showSecret mode

Note: Currently ignored under WinCE.

Remarks

Returns `NULL` if the database does not exist or fails to open properly.

Returns

The named `AGDBSet` object.

remove( )

Deletes the `AGDBSet` object named `dbname`.

Interface

`AGDBDatabaseManager`

IDL definition

void remove(const String *dbname*);

ADO equivalent

None [

JavaScript synopsis

dbmgr.**remove(***dbname***)**

C synopsis

```
void AVremove(
  AGDBDatabaseManager* dbmgr,
  const PODSString dbname
);
```

Parameters

- ***dbmgr*** The `AGDBDatabaseManager` object.
- ***dbname*** [in] The name of the database to delete.

Remarks

Use "exists( )" on page 384 to determine whether the `AGDBSet` object was successfully deleted.

Returns

None

types

Gets the `AGDBColumnTypes` object.

Interface

`AGDBDatabaseManager`

IDL definition

readonly attribute AGDBColumnTypes types;

ADO equivalent

None

JavaScript synopsis

dbmgr.**types**

C synopsis

AGDBColumnTypes* **AVgetTypes(**AGDBDatabaseManager* *dbmgr***);**

Parameters

- ***dbmgr*** The `AGDBDatabaseManager` object.

Returns

The `AGDBColumnTypes` object.

AGDBSearch object

- **Inherits from:** `AGDBObject`
- **Available to:** JavaScript and C

The `AGDBSearch` object is returned by the "createSearch( )" on page 339 method on the `AGDBSet` object. The "find( )" on page 343 method uses the `AGDBSearch` object.

The `AGDBSearch` object is an opaque object, having no methods or attributes of its own.

AGDBBlob object

- **Inherits from:** AGDBObject
- **Available to:** C only

The AGDBBlob object holds blob data from an AGDBSet object.

Summary of AGDBBlob attributes and methods

Description	Attributes and methods
Accessing information about data in the AGDBBlob object	"getBlobData( )" on page 389 "get_element( )" on page 390 "length" on page 390 "set_element( )" on page 391

getBlobData( )

Returns a pointer to the data inside a blob.

Interface

AGDBBlob

IDL definition

nometadata AGDBByte getBlobData();

ADO equivalent

None

JavaScript synopsis

Not applicable

C synopsis

AGDBByte* **AVgetBlobData(**AGDBBlob* *blob*);

Parameters

- ***blob*** The AGDBBlob object.

Returns

A pointer to the data inside a blob.

get_element( )

Retrieves a single byte from the blob data.

Interface

AGDBBlob

IDL definition

```
nometadata void get_element(
  unsigned long index,
  PODSVariant variant
);
```

ADO equivalent

None

JavaScript synopsis

blob[*index*]

C synopsis

```
void AVget_element(
  AGDBBlob* blob,
  PODSUInt32 index,
  PODSVariant* variant
);
```

Parameters

- ***blob*** The AGDBBlob object.
- ***index*** [in] The index of the byte to retrieve.
- ***variant*** [in] The variant into which to return the byte.

Returns

None

See also

"set_element( )" on page 391

length

Returns the length of the blob data.

Interface

AGDBBlob

IDL definition

```
attribute unsigned long length;
```


ADO equivalent

None

JavaScript synopsis

blob.**length**

blob.**length =** *length*

C synopsis

PODSUInt32 **AVgetLength(**AGDBBlob* *blob***);**

void **AVsetLength(**
AGDBBlob* *blob***,**
PODSUInt32 *length*
);

Parameters

- ***blob*** The `AGDBBlob` object.
- ***length*** [in] The length of the specified blob data.

Returns

Getter: The length of the blob data.

Setter: None

set_element( )

Sets a single byte in the blob data.

Interface

`AGDBBlob`

IDL definition

nometadata void set_element(
unsigned long *index*,
PODSVariant *variant*
);

ADO equivalent

None

JavaScript synopsis

blob[*index*] = *variant*

C synopsis

void **AVset_element(**
AGDBBlob* *blob***,**
PODSUInt32 *index***,**


```
  PODSVariant* variant
);
```

Parameters

- ***blob*** The `AGDBBlob` object.
- ***index*** [in] The index of the byte to set.
- ***variant*** [in] The variant containing the value to assign to the byte.

Returns

None

CHAPTER 14

M-Business SOAP API reference

Contents

Overview

The M-Business SOAP API allows developers to programmatically execute functions that would otherwise have to be performed through the M-Business Administration Console user interface. This API is very different from all the other APIs documented in this guide:

- The program you write to access the M-Business SOAP API runs on the desktop, not on M-Business Client. This program may be referred to as a SOAP client, which is totally unrelated to M-Business Client.
- You may access functions in the M-Business SOAP API from any language that supports SOAP; the sample code for Java and C# ships with M-Business Server.
- You may not programmatically initiate a synchronization from a SOAP client.
- To trap errors returned by SOAP functions, you first check for an HTTP 200 error, then compare the value of the `FaultString` parameter with a list of error conditions. See "M-Business SOAP API error messages" on page 396.

Using the M-Business SOAP API functions

To use the M-Business SOAP API, the SOAP client must first successfully call `loginUser()`.The SOAP client must then retrieve the `sessionId` in the response header and send it with all subsequent function calls. Refer to the Java and C# samples we ship to see how to do this. For an overview, see Appendix "SOAP sample client files" on page 497.

Permissions

All functions are available to the built-in `admin` user. A regular user may call a subset of these functions. If you see a `permission denied` error, you must be `admin` to call that function.

Error messages

Each SOAP client must use the mechanism in its SOAP client library to handle the response from each function call. If using SOAP over HTTP, a `200` response from the M-Business SOAP Server indicates the function succeeded. If not, a SOAP-formatted error message is returned. Your SOAP client should trap and act on these. When developing, the M-Business SOAP Server has several options for logging SOAP messages. Refer to the *agsoap.conf* file for more information.

Pseudocode

Because you call a SOAP function in very different ways from each language, the reference material in this document provides pseudocode instead of a synopsis.

Caution on 'INTERNAL' functions

Many functions are marked INTERNAL. In future releases of the M-Business SOAP API, these functions may be modified in a way that is not backward compatible with the functions in this release, or these functions may not be supported at all. If you use a function marked INTERNAL, you may have to rewrite your application when you upgrade your M-Business Server.

About the code samples

M-Business Server ships with the source code to a complete sample SOAP client, with source files for Java and C#. Start by reviewing Appendix "SOAP sample client files" on page 497.

Refer to the Java or C# samples provided for language-specific details. These sample files are located in the *<M-Business_Home>/samples* directory on the server where M-Business Server is installed. You execute the sample code using the directions provided in Appendix "SOAP sample client files" on page 497.

M-Business SOAP API error messages

Each SOAP client must use the mechanism in its SOAP client library to handle the response from each function call. If you use SOAP over HTTP, a `200` response from the SOAP server indicates the function succeeded. If not, a SOAP-formatted error message is returned ("M-Business SOAP API error messages" on page 396). Your SOAP client should trap and act on these. When developing, the M-Business SOAP Server has several options for logging SOAP messages. Refer to `agsoap.conf` for more information.

Use the `Faultstring` parameter when a SOAP function returns a formatted error message to determine the cause of the error. First, convert the `Faultstring` to lower case text, then compare it to the following list of error messages.

Table 1. SOAP error message Faultstring contents

If `Faultstring` parameter contains:	Error is:
`ag_firstname_too_long`	The user's first name is too long.
`ag_group_exists`	The group name you are trying to create already exists.
`ag_lastname_too_long`	The user's last name is too long.
`ag_license_error`	There was a licensing problem. Most likely you are trying to add a user beyond your licensed number of users.
`ag_password_too_long`	The user's password is too long.
`ag_user_exists`	The user name you are trying to create already exists.
`ag_user_old_password_mismatch`	The user's old password has not been matched when attempting to change the user's password.
`ag_username_too_long`	The `username` is too long.
`agp_category_exists`	The public channel category name you are trying to create already exists.
`agp_xmldb_exists`	The XML database name you are trying to create already exists.

If `Faultstring` parameter contains:	Error is:
`java.net.ConnectException`	The M-Business SOAP Server might not be running, or you may be trying to connect to the wrong host name or port. Note: You only see this error if you are using a Java SOAP client,
`verify and session`	The current session has expired, you have to call `loginUser` again.

Summary of M-Business SOAP API functions

Refer to the following table to view a list of M-Business SOAP API functions, organized by the type of M-Business Server functionality they are used to manage.

Note
Order in which function groups are presented is based on frequency with which they are likely to be used. Functions within a group appear in alphabetical order.

Functionality	Associated function(s)
“Session management functions” on page 403	“loginUser( )” on page 403 “logoutUser( )” on page 403
“User management functions” on page 405	“userChangePassword( )” on page 405 “userClearCache( )” on page 405 “userCreate( )” on page 406 “userDelete( )” on page 406 “userFindUsers( )” on page 406 “userGetBasicInfo( )” on page 407 “userGetGroups( )” on page 407 “userGetInfo( )” on page 408 “userGetSyncLogs( )” on page 408 “userGetWebConduitSyncState( )” on page 408 “userSetWebConduitSyncState( )” on page 409 “userUpdate( )” on page 409

Functionality	Associated function(s)
"Group management functions" on page 414	"groupGetAll( )" on page 416 "groupGetAllListInfo( )" on page 416 "groupGetInfo( )" on page 416 "groupCreate( )" on page 415 "groupDelete( )" on page 415 "groupUpdate( )" on page 419 "groupAddXmlChannel( )" on page 415 "groupRemoveXmlChannel( )" on page 418 "groupUpdateXmlChannel( )" on page 419 "groupGetXmlChannelDetail( )" on page 417 "groupAddUser( )" on page 414 "groupRemoveUser( )" on page 418 "groupGetGroupsForUser( )" on page 416 "groupAddAllUsersIntoGroup( )" on page 414 "groupAddAdmin( )" on page 414 "groupRemoveAdmin( )" on page 417 "groupIsUserGroupAdmin( )" on page 417
"Web channel functions" on page 425	"webchannelCreate( )" on page 425 "webchannelDelete( )" on page 425 "webchannelGetAll( )" on page 426 "webchannelGetInfo( )" on page 426 "webchannelGetSynced( )" on page 427 "webchannelUpdate( )" on page 427

Functionality	Associated function(s)
“Public channel functions” on page 433	“webchannelCreateCategory( )” on page 433 “webchannelDeleteCategory( )” on page 433 “webchannelDeleteCategoryChannel( )” on page 433 “webchannelFindPublicChannels( )” on page 434 “webchannelGetAllCategories( )” on page 434 “webchannelGetCategoryChannelCount( )” on page 434 “webchannelGetCategoryInfo( )” on page 435 “webchannelGetPublicChannelIds( )” on page 435 “webchannelMoveCategoryToTopLevel( )” on page 435 “webchannelSubscribeToPublicChannel( )” on page 436 “webchannelUnsubscribeFromPublicChannel( )” on page 436 “webchannelUpdateCategory( )” on page 437
“Report functions” on page 440	“sqlQuery( )” on page 440
“M-Business Server configuration functions” on page 442	“configAllowSelfRegistration( )” on page 442 “configEnableConduit( )” on page 443 “configEnablePersonalChannels( )” on page 444 “configEnableSelfRegistration( )” on page 444 “configGetInfo( )” on page 445 “configSetAdminPassword( )” on page 445 “configSetMinPasswordLength( )” on page 445

Summary of M-Business SOAP API data structures

The following table of M-Business SOAP API data structures is organized by M-Business Server function. In most cases, these data structures are used only by the corresponding group of functions.

Note
Each M-Business SOAP API data structure group is presented immediately after the corresponding function group. Within a data structure group, individual data structures are presented in alphabetical order.

Functional group	Associated data structures
"Session management data structures" on page 403	"LoginUserResponse" on page 404
"User management data structures" on page 409	"BasicUserDetail" on page 410 "getFindUsersResponse" on page 410 "NewUser" on page 411 "SyncLog" on page 411 "User" on page 412 "UserDetail" on page 412
"Group management data structures" on page 419	"Group" on page 420 "GroupDetail" on page 420 "GroupListInfo" on page 421 "GroupUserListItem" on page 421 "NewGroup" on page 422
"Database (XML) channel data structures" on page 422	"NewXmlChannel" on page 423 "XmlChannel" on page 423 "XmlChannelDetail" on page 424
"Web channel data structures" on page 428	"NewWebChannel" on page 428 "SyncedWebChannel" on page 429 "WebChannel" on page 430 "WebChannelDetail" on page 430

Functional group	Associated data structures
“Public channel data structures” on page 437	“BasicPublicChannel” on page 437 “Category” on page 438 “CategoryChannelCount” on page 438 “NewCategory” on page 439 “PublicChannelId” on page 439
“Report data structures” on page 440	“Resultset” on page 440
“M-Business Server configuration data structures” on page 446	“ServerConfig” on page 446

Session management functions

A SOAP client must successfully login before making any other API calls. The `login` function returns a session ID in the SOAP header. This session ID must be sent in the header of all subsequent function calls.

loginUser( )

Authenticates a user and start a session.

Pseudocode

```
loginUser(
  string userName,
  string password
);
```

Returns

The user ID in a `LoginUserResponse` structure.

Parameters

- ***userName*** The user name of the person logging in.
- ***password*** The password of user logging in.

Remarks

The password parameter must be blowfish encrypted and base 64 encoded. You can find the `AuthKey` you need to blowfish the password in `conf/agsoap.conf`.

logoutUser( )

Ends a user session.

Pseudocode

```
logoutUser(string userName);
```

Parameters

- ***userName*** The user name of the person being logged out.

Returns

None

Session management data structures

Session management data structures are used to initiate a session with the user.

LoginUserResponse

The unique user ID returned by the function `loginUser()`.

Structure

```
{
   unsigned int userId;
};
```

Fields

- ***userId*** The unique ID of this user.

Remarks

`info.infoId = 0` inserts, otherwise it updates. INTERNAL - In future releases, may not be backward compatible or may not be supported at all.

User management functions

Use the following functions to manage your users on M-Business Server.

userChangePassword( )

Changes a user's password.

Pseudocode

```
userChangePassword(
  unsigned int userId,
  string oldPassword,
  string newPassword
);
```

Parameters

- ***userId*** The unique user ID.
- ***oldPassword*** The user's current password.
- ***newPassword*** The user's new password.

Returns

None

userClearCache( )

Clears any combination of caches for an individual user.

Pseudocode

```
userClearCache(
  unsigned int userId,
  boolean cache,
  boolean cookie,
  boolean auth
);
```

Parameters

- ***userId*** The unique user ID.
- ***cache*** Clear the actual cache.
- ***cookie*** Clear all cookies.
- ***auth*** Clear all authentication information.

Returns

None

userCreate( )

Creates a new user account.

Pseudocode

userCreate(NewUser *user*);

Parameters

- ***user*** The unique user ID.

Returns

The new unique `unsigned int` user ID.

Remarks

May be called without being logged in.

userDelete( )

Deletes a user account.

Pseudocode

userDelete(unsigned int *userId*);

Parameters

- ***userId*** The unique user ID.

Returns

None

userFindUsers( )

Finds users by name.

Pseudocode

```
userFindUsers(
  string userName,
  string firstName,
  string lastName,
  unsigned int startPosition,
  unsigned int maxResults,
  unsigned int groupId,
  boolean usersNotInGroup,
  boolean adminsNotInGroup
);
```

Parameters

- ***userName*** Search by user name.

- ***firstName*** Search by a user's first name.
- ***lastName*** Search by a user's last name.
- ***startPosition*** Starting index in the database.
- ***maxResults*** Maximum number of results returned.
- ***groupId*** An optional group id.
- ***usersNotInGroup*** Search for users not in any group.
- ***adminsNotInGroup*** Search for administrators not in any group.

Returns

The information in a `getFindUsersResponse` structure.

Remarks

May be called multiple times to get results in chunks, by using the `startPosition` index. Also allows searching for users who are not in a group or are not group administrators.

userGetBasicInfo( )

Retrieves some basic user information.

Pseudocode

userGetBasicInfo(unsigned int *userId*);

Parameters

- ***userId*** The unique user ID.

Returns

The information in a `BasicUserDetail` structure.

userGetGroups( )

Retrieves a list of groups in which the user is a member.

Pseudocode

userGetGroups(unsigned int *userId*);

Parameters

- ***userId*** The unique user ID.

Returns

The information in a `GroupArray` array.

userGetInfo( )

Retrieves detailed user information.

Pseudocode

userGetInfo(unsigned int *userId*);

Parameters

- ***userId*** The unique user ID.

Returns

The information in a `UserDetail` structure.

userGetSyncLogs( )

Retrieves the synchronization logs for a user.

Pseudocode

```
userGetSyncLogs(
  unsigned int userId,
  unsigned int count
);
```

Parameters

- ***userId*** The unique user ID.
- ***count*** The number of times the user has synchronized a device.

Returns

The information in a `SyncLogArray` array.

Remarks

INTERNAL - In future releases, may not be backward compatible or may not be supported at all.

userGetWebConduitSyncState( )

Determines whether the user will synchronize web content.

Pseudocode

userGetWebConduitSyncState(unsigned int *userId*);

Parameters

- ***userId*** The unique user ID.

Returns

Boolean value

userSetWebConduitSyncState( )

Turns web content synchronization on or off for an individual user.

Pseudocode

```
userSetWebConduitSyncState(
  unsigned int userId,
  boolean sync
);
```

Parameters

- ***userId*** The unique user ID.
- ***sync*** Set to `true` to synchronize web content, `false` to do nothing.

Returns

None

userUpdate( )

Modifies a user account.

Pseudocode

```
userUpdate(
  unsigned int userId,
  NewUser user
);
```

Parameters

- ***userId*** The unique user ID.
- ***user*** The new user ID.

Returns

None

User management data structures

User management functions are used to manage user information stored in the M-Business Server database. Users must have an account in order to access M-Business Server. M-Business Server has three types of users:

- Administrator: (admin) is a built-in account for managing M-Business Server.
- User: Anyone who synchronizes with the server using M-Business Client.
- Group Administrator: A regular user who can also manage one or more groups.

If M-Business Server is integrated with an external authentication server (NT Domain, LDAP), then some of these functions will not apply (you will get a SOAP error). See the configuration functions to detect this.

BasicUserDetail

Basic information about a user. `info.infoId = 0` inserts, otherwise it updates.

Structure

```
{
    string userName;
    string firstName;
    string lastName;
    string comment;
};
```

Fields

- ***userName*** The unique user (login) name.
- ***firstName*** The user's first name.
- ***lastName*** The user's last name.
- ***comment*** An optional comment about the user.

Remarks

The information is an array of `User` structures. INTERNAL - In future releases, may not be backward compatible or may not be supported at all.

getFindUsersResponse

Returned by the `userFindUsers( )` function.

Structure

```
{
    unsigned int endPosition;
    unsigned int totalCount;
    unsigned int matchCount;
    UserArray users;
};
```

Fields

- ***endPosition*** The end position of the index.
- ***totalCount*** The total user count.
- ***matchCount*** The search match count.
- ***users*** The list of users.

Remarks

The information is an array of `User` structures.

NewUser

Creates a new user. The `userName` parameter can be an email address. All fields have a size limit of 64 characters.

Structure

```
{
    string userName;
    string firstName;
    string lastName;
    string password;
    string comment;
};
```

Fields

- ***userName*** The unique user (login) name.
- ***firstName*** The user's first name.
- ***lastName*** The user's last name.
- ***password*** The user's password.
- ***comment*** An optional comment about the user.

Remarks

INTERNAL - In future releases, may not be backward compatible or may not be supported at all.

SyncLog

Contains synchronization log entries for a user.

Structure

```
{
    long startTime;
    long endTime;
    unsigned int seconds;
    string errorMessage;
};
```

Fields

- ***startTime*** The synchronization start time.
- ***endTime*** The synchronization end time.
- ***seconds*** The duration of the synchronization in seconds.

- ***errorMessage*** Generated error messages. INTERNAL - In future releases, may not be backward compatible or may not be supported at all.

Returns

The information is an array of `SyncLog` structures.

User

Basic information for listing users

Structure

```
{
    unsigned int userId;
    string userName;
    string firstName;
    string lastName;
    string deviceOS;
};
```

Fields

- ***userId*** The unique user ID.
- ***userName*** The unique user (login) name.
- ***firstName*** The user's first name.
- ***lastName*** The user's last name.
- ***deviceOS*** The last O/S synchronized by the user.

Returns

The information is an array of `User` structures.

Remarks

`info.infoId = 0` inserts, otherwise it updates. INTERNAL - In future releases, may not be backward compatible or may not be supported at all.

UserDetail

The full description of a user.

Structure

```
{
    unsigned int userId;
    string userName;
    string firstName;
    string lastName;
    string password;
    string comment;
```



```
    long modifiedTime;
    long createdTime;
    long syncStartTime;
    boolean isAuthHashed;
    WebChannelArray webChannelArray;
    GroupArray groupArray;
};
```

Fields

- ***userId*** The unique ID of the user.
- ***userName*** The unique user name.
- ***firstName*** The user's first name.
- ***lastName*** The user's last name.
- ***password*** The user's password.
- ***comment*** An optional comment about the user.
- ***modifiedTime*** The time the user information was last modified.
- ***createdTime*** The time user was created.
- ***syncStartTime*** The time the user last started a synchronization.
- ***isAuthHashed*** Indicates whether the client password is hashed. INTERNAL - In future releases, may not be backward compatible or may not be supported at all.
- ***webChannelArray*** A list of channels to which the user is subscribed.
- ***groupArray*** A list of groups to which the user is subscribed.

Remarks

The information is an array of `User` structures. `info.infoId = 0` inserts, otherwise it updates. INTERNAL - In future releases, may not be backward compatible or may not be supported at all.

Group management functions

Use the following functions to manage your groups.

groupAddAdmin( )

Makes an existing user an administrator of a group.

Pseudocode

```
groupAddAdmin(
  unsigned int groupId
  unsigned int userId
);
```

Parameters

- ***groupId*** The unique ID of this group.
- ***userId*** The unique ID of this user.

Returns

None

groupAddAllUsersIntoGroup( )

Adds all existing users into a group.

Pseudocode

```
groupAddAllUsersIntoGroup(unsigned int groupId);
```

Parameters

- ***groupId*** The unique ID of this group.

Returns

None

groupAddUser( )

Adds an existing user to a group.

Pseudocode

```
groupAddUser(
  unsigned int groupId,
  unsigned int userId
);
```


Parameters

- ***groupId*** The unique ID of this group.
- ***userId*** The unique ID of this user.

Returns

None

groupAddXmlChannel( )

Adds a database (XML) channel to a group.

Pseudocode

```
groupAddXmlChannel(
  unsigned int groupId,
  NewXmlChannel channel
);
```

Parameters

- ***groupId*** The unique ID of this group.
- ***channel*** The unique ID of this database channel.

Returns

The new unique `unsigned int` channel ID.

groupCreate( )

Creates a new, unique `unsigned int` group ID.

Pseudocode

groupCreate(NewGroup *group*);

Parameters

- ***group*** The new, unique `unsigned int` group ID.

Returns

The new, unique `unsigned int` group ID.

groupDelete( )

Deletes a group.

Pseudocode

groupDelete(unsigned int *groupId*);

Parameters

- ***groupId*** The unique `unsigned int` group ID of the group to be deleted.

Returns

None

groupGetAll( )

Retrieves a list of basic information about all groups.

Pseudocode

groupGetAll( );

Returns

The information in a `GroupArray` array.

groupGetAllListInfo( )

Retrieves a list of information for display about all groups.

Pseudocode

groupGetAll( );

Returns

The information in a `GroupListInfoArray` array.

groupGetGroupsForUser( )

Retrieves a list of all groups of which the user is a member.

Pseudocode

groupGetGroupsForUser(unsigned int *userId*);

Parameters

- ***userId*** The unique user ID.

Returns

The information in a `GroupUserListItemArray` array.

groupGetInfo( )

Retrieves detailed information about a group.

Pseudocode

groupGetInfo(unsigned int *groupId*);

Parameters

- ***groupId*** The unique group ID.

Returns

The information in a `GroupDetail` structure.

groupGetXmlChannelDetail( )

Retrieves the details about a database (XML) channel in a group.

Pseudocode

```
groupGetXmlChannelDetail(
  unsigned int groupId,
  unsigned int channelId
);
```

Parameters

- ***groupId*** The unique group ID.
- ***channelId*** The unique channel ID.

Returns

The information in a `XmlChannelDetail` structure.

groupIsUserGroupAdmin( )

Determines whether the user is an administrator of any group.

Pseudocode

groupIsUserGroupAdmin(unsigned int *userId*);

Parameters

- ***userId*** The unique user ID.

Returns

A boolean value.

groupRemoveAdmin( )

Removes a user from the administrator list of a group.

Pseudocode

```
groupAddAdmin(
  unsigned int groupId
  unsigned int userId
);
```

Parameters

- ***groupId*** The unique group ID.
- ***userId*** The unique user ID.

Returns

None

groupRemoveUser( )

Removes a user from the membership list of a group.

Pseudocode

```
groupRemoveUser(
  unsigned int groupId,
  unsigned int userId
);
```

Parameters

- ***groupId*** The unique group ID.
- ***userId*** The unique user ID.

Returns

None

groupRemoveXmlChannel( )

Removes a database (XML) channel from a group.

Pseudocode

```
groupRemoveXmlChannel(
  unsigned int groupId,
  unsigned int channelID
);
```

Parameters

- ***groupId*** The unique group ID.
- ***channelId*** The unique channel ID.

Returns

None

groupUpdate( )

Modifies a group.

Pseudocode

```
groupUpdate(
  unsigned int groupId,
  NewGroup group
);
```

Parameters

- ***groupId*** The unique group ID.
- ***group*** The new unique group ID.

Returns

None

groupUpdateXmlChannel( )

Modifies a database (XML) channel in a group.

Pseudocode

```
groupUpdateXmlChannel(
  unsigned int groupId,
  unsigned int channelID,
  NewXmlChannel channel
);
```

Parameters

- ***groupId*** The unique group ID.
- ***channelID*** The unique database channel ID.
- ***channel*** The name of the database channel.

Returns

None

Group management data structures

Groups are used as convenient holders for web and database channels. Administrators (or group administrators) can manage a group and determine which users are in it (membership).

Group type constants

Supported group types and their associated values are as follows:

```
{
     PRIVATE_GROUP,    // 0
     PUBLIC_GROUP,     // 1
     MANDATORY_GROUP   // 2
};
```

For a detailed description of these group types, see "Understanding group types" [*M-Business Anywhere Administrator Guide*].

Group

Basic group information.

Structure

{
  unsigned int *groupId*;
  string *name*;
  string *description*;
};

Fields

- ***groupId*** Unique ID of this group.
- ***name*** Unique group name.
- ***description*** Group description.

Remarks

`GroupArray` is an array of `group` structures.

GroupDetail

The full description of a `group`.

Structure

{
  string *name*;
  string *description*;
  boolean *isDeleted*;
  unsigned int *type*;
  WebChannelArray *webChannelArray*;
  xmlChannelArray *xmlChannelArray*;
  UserArray *userArray*;
  UserArray *adminArray*;
};

Fields

- ***name*** Unique group name.
- ***description*** Group description.
- ***isDeleted*** Has this group been deleted?
- ***type*** One of `GroupType`.
- ***webChannelArray*** Web channels of group.
- ***xmlChannelArray*** Database channels of group.
- ***userArray*** Users of group.
- ***adminArray*** Administrators of group.

GroupListInfo

Information for listing groups.

Structure

```
{
    unsigned int id;
    unsigned int userCount;
    string name;
    string description;
    unsigned int typeInt;
};
```

Fields

- ***id*** Unique ID of this group.
- ***userCount*** Number of users in group.
- ***name*** Unique group name.
- ***description*** Group description.
- ***typeInt*** One of `GroupType`.

Remarks

`GroupListInfoArray` is an array of `GroupListInfo` structures.

GroupUserListItem

Detailed information about a group to which a user belongs.

Structure

```
{
    unsigned int uid;
```

```
    unsigned int gid;
    string name;
    string description;
    boolean isMember;
    boolean isAdmin;
};
```

Fields

- ***uid*** Unique ID of user.
- ***gid*** Unique ID of group.
- ***name*** Unique group name.
- ***description*** Group description.
- ***isMember*** Is user a member of group?
- ***isAdmin*** Is user an admin of group?

Remarks

`GroupUserListItemArray` is an array of `GroupUserListItem` structures.

NewGroup

Used to create or update a group.

Structure

```
{
    string name;
    string description;
    unsigned int type;
};
```

Fields

- ***name*** Unique group name.
- ***description*** Group description.
- ***type*** One of `GroupType`.

Database (XML) channel data structures

Database channels are used to send XML data (and schema) to the device. Database channels can only be defined for groups.

> **Note**
> Devices can perform form posts with only changed data. For more information, see "Performing incremental synchronizations" [*M-Business Anywhere Application Developer Guide*].

NewXmlChannel

Creates/updates a database channel.

Structure

```
{
    string dbName;
    string dataUrl;
    string schemaUrl;
    string deleteUrl;
    unsigned int dataFormat;
    boolean fetchModsOnly;
    boolean fetchDeleteType;
};
```

Fields

- ***dbName*** Name of the on-device database.
- ***dataUrl*** URL from which to retrieve XML data.
- ***schemaUrl*** URL from which to retrieve XSD schema.
- ***deleteUrl*** URL to retrieve when deletes occur.
- ***dataFormat*** `1` = attribute

  `2` = element
- ***fetchModsOnly*** Set to `true` to update only changes, `false` to update all data.
- ***fetchDeleteType*** `0` = implicit

  `1` = explicit

Remarks

`PublicChannelIdArray` is an array of `PublicChannelId` structures.

XmlChannel

The basic database channel information required to display a list of channels.

Structure

```
{
    unsigned int channelId;
    string dbName;
    long modifiedTime;
};
```

Fields

- ***channelId*** The unique ID of this channel.
- ***dbName*** The name of the on-device database.

- ***modifiedTime*** The last channel modified timestamp.

Remarks

`XmlChannelArray` is an array of `XmlChannel` structures.

XmlChannelDetail

The full description of a database channel.

Structure

```
{
    unsigned int id;
    string dbName;
    string dataUrl;
    string schemaUrl;
    string deleteUrl;
    unsigned int dataFormat;
    boolean fetchModsOnly;
    boolean fetchDeleteType;
};
```

Fields

- ***id*** The unique channel ID.
- ***dbName*** The name of the on-device database.
- ***dataUrl*** The URL from which to fetch XML data.
- ***schemaUrl*** The URL from which to fetch the XSD schema.
- ***deleteUrl*** The URL to fetch when deletes occur.
- ***dataFormat*** `1` = attribute

  `2` = element
- ***fetchModsOnly*** Set to `true` to update only changes, `false` to update all data.
- ***fetchDeleteType*** `0` = implicit

  `1` = explicit

Web channel functions

Use the following functions to manage your web channels on M-Business Server.

webchannelCreate( )

Creates a web channel.

Pseudocode

```
webchannelGetInfo(
  unsigned int chanType,
  unsigned int refUserOrGroupId,
  NewWebChannel webChannel
);
```

Returns

The new, unique`unsigned int` web channel ID.

Parameters

- ***chanType*** The type of channel.
- ***refUserOrGroupId*** The unique ID of the user or group.
- ***webChannel*** The name of the new web channel.

Remarks

The `refUserOrGroupId` parameter must be set to an existing user or group ID, depending on `chanType`.

webchannelDelete( )

Deletes a web channel by channel ID.

Pseudocode

```
webchannelDelete(
  unsigned int chanType,
  unsigned int refUserOrGroupId,
  unsigned int ChannelId
);
```

Parameters

- ***chanType*** The type of channel.
- ***refUserOrGroupId*** The unique ID of the user or group.
- ***ChannelId*** The unique ID of the new web channel.

Returns

None

Remarks

The `refUserOrGroupId` parameter must be set to an existing user or group ID, depending on `chanType`.

webchannelGetAll( )

Retrieves a list of all web channels of one type, by user or group ID.

Pseudocode

```
webchannelGetAll(
  unsigned int chanType,
  unsigned int refUserOrGroupId
);
```

Parameters

- ***chanType*** The type of channel.
- ***refUserOrGroupId*** The unique ID of the user or group.

Returns

A `WebChannelArray`array.

Remarks

The `refUserOrGroupId` parameter must be set to an existing user or group ID, depending on `chanType`.

webchannelGetInfo( )

Retrieves detailed information about one web channel,

Pseudocode

```
webchannelGetInfo(
  unsigned int chanType,
  unsigned int refUserOrGroupId,
  unsigned int channelId
);
```

Parameters

- ***chanType*** The type of channel.
- ***refUserOrGroupId*** The unique ID of the user or group.
- ***ChannelId*** The unique ID of the web channel.

Returns

The information in a `WebChannelDetail` structure.

Remarks

The `refUserOrGroupId` parameter must be set to an existing user or group ID, depending on `chanType`.

webchannelGetSynced( )

Retrieves a list of channels that have been synchronized to a user's device.

Pseudocode

webchannelDelete(unsigned int *userId*);

Parameters

- ***userId*** The unique user ID.

Returns

The information in a `SyncedWebChannelArray` array.

webchannelUpdate( )

Modifies a web channel.

Pseudocode

```
webchannelUpdate(
  unsigned int chanType,
  unsigned int refUserOrGroupId,
  unsigned int ChannelId,
  newWebChannel webChannel
);
```

Parameters

- ***chanType*** The type of channel.
- ***refUserOrGroupId*** The unique ID of the user or group.
- ***ChannelId*** The unique ID of the web channel.
- ***webChannel*** The name of the updated web channel.

Returns

None

Remarks

The `refUserOrGroupId` parameter must be set to an existing user or group ID, depending on `chanType`.

Web channel data structures

A web channel is an URL that we can synchronize to a device. These types of channels are used by users: for example, personal channels.

Channel type constants

Supported channel types and their associated values are as follows:

```
{
    USER_CHANNEL,      //  0
    GROUP_CHANNEL,   //  1
    PUBLIC_CHANNEL    //  2
};
```

NewWebChannel

Use to create/update a web channel.

Structure

```
{
    unsigned int chanType;
    unsigned int refUserOrGroupId;
    string title;
    string url;
    unsigned int maxSize;
    unsigned int linkDepth;
    boolean allowImages;
    boolean offsitelinks;
    string refreshBehaviour;
    unsigned int hours;
    unsigned int wdo;
    string rtime;
    unsigned int flags;
    boolean allowBinaryDistribution;
    boolean isHidden;
    unsigned int colorDepth;
    string description;
    unsigned int category;
    boolean syncOnlyChannel;
    string startSyncUrl;
    string endSyncUrl;
};
```

Fields

- ***chanType*** The type of channel, selected from one of *ChannelType*.
- ***refUserOrGroupId*** The user or group ID of channel's owner.
- ***title*** The channel's display name.

- ***url*** The channel's URL.
- ***maxSize*** The channel's maximum content size: Kbyte.
- ***linkDepth*** The depth of linked pages to retrieve.
- ***allowImages*** Shows channel images on device when selected.
- ***offsitelinks*** Follows offsite channel links when selected.
- ***refreshBehaviour*** Sets the channel content refresh schedule: values are *once*, *always*, *hourly*, and *daily*.
- ***hours*** The hour interval when the refresh behavior is set to *hourly*.
- ***wdo*** Refresh channel content on week days only: 0 = no, 1= refresh.
- ***rtime*** The time of day to refresh channel content when the refresh schedule is set to *daily*.
- ***flags*** Bit flags indicating the days of the week on which to refresh channel content.
- ***allowBinaryDistribution*** Flag, when set, allows the distribution of binary files.
- ***isHidden*** Flag, when set, identifies the channel as hidden.
- ***colorDepth*** The color depth at which to display channel content: 0, 1, 2, 4, 8, and 16.
- ***description*** A description of the public channel.
- ***category*** The category of the public channel.
- ***syncOnlyChannel*** Flag, when set, determines that the channel will only update during a synchronization and not go online.
- ***startSyncUrl*** The first URL to retrieve as the synchronization starts.
- ***endSyncUrl*** The last URL to retrieve when synchronization ends.

Remarks

`WebChannelArray` is an array of `WebChannel` structures.

SyncedWebChannel

Describes a list of channels that have been synchronized to a device.

Structure

```
{
    unsigned int id;
    unsigned int size;
    string lastUpdated;
    string syncTime;
};
```


Fields

- ***id*** The channel's unique ID.
- ***size*** The channel's size, in bytes.
- ***lastUpdated*** The channel's last update time.
- ***syncTime*** The channel's last synchronization time.

Remarks

`SyncedWebChannelArray` is an array of `SyncedWebChannel` structures.

WebChannel

The basic web channel information required to display a list of channels.

Structure

```
{
    unsigned int channelId;
    unsigned int chanType;
    unsigned int refUserOrGroupId;
    boolean isHidden;
    string title;
    string url;
};
```

Fields

- ***channelId*** The channel's unique ID.
- ***chanType*** The channel's type.
- ***refUserOrGroupId*** The user or group ID of the channel owner.
- ***isHidden*** Indicates whether the channel is hidden.
- ***title*** The name of the channel.
- ***url*** The URL of the channel

Remarks

`WebChannelArray` is an array of `WebChannel` structures.

WebChannelDetail

The full description of a web channel.

Structure

```
{
    string title;
    string url;
```



```
    unsigned int maxSize;
    unsigned int linkDepth;
    boolean allowImages;
    boolean offsitelinks;
    string refreshBehaviour;
    unsigned int hours;
    unsigned int wdo;
    string rtime;
    unsigned intflags;
    long modifiedTime;
    boolean allowBinaryDistribution;
    boolean isHidden;
    unsigned int colorDepth;
    string description;
    unsigned int category;
    boolean syncOnlyChannel;
    string startSyncUrl;
    string endSyncUrl;
};
```

Fields

- ***title*** The channel's display name.
- ***url*** The channel's URL.
- ***maxSize*** The maximum size of channel content in Kbytes.
- ***linkDepth*** The link depth of pages to retrieve.
- ***allowImages*** Does channel display images on-device?
- ***offsitelinks*** Follow offsite links?
- ***refreshBehaviour*** The refresh behavior of the channel: *once*, *always*, *hourly*, or *daily*.
- ***hours*** channel refresh schedule is hourly: *0* , *1*.
- ***wdo*** channel refresh schedule is weekdays only: *0* , *1*.
- ***rtime*** channel refresh schedule is `refresh=daily`, `refresh time`
- ***flags*** Bit flags for refresh days.
- ***modifiedTime*** channel last modified time.
- ***allowBinaryDistribution*** Bit flag allows binary distribution of files.
- ***isHidden*** Bit flag indicates whether the channel is hidden.
- ***colorDepth*** The color depth at which to display channel content: *0* , *1*, *2*, *4*, *8*, and *16*.
- ***description*** The public channel description.
- ***category*** The public channel category.
- ***syncOnlyChannel*** Flag determines whether a channel can only perform a synchronization to retrieve data (not go online).

- ***startSyncUrl*** The first URL to retrieve as the synchronization starts.
- ***endSyncUrl*** The last URL retrieved when the synchronization ends.

Remarks

`WebChannelDetailArray` is an array of `WebChannelDetail` structures.

Public channel functions

Use the following functions to manage your public channels.

webchannelCreateCategory( )

Creates a new web channel category.

Pseudocode

webchannelCreateCategory(NewCategory *category*);

Parameters

- ***category*** The new web channel category ID.

Returns

The new, unique `unsigned int` category ID.

webchannelDeleteCategory( )

Deletes an existing web channel category.

Pseudocode

webchannelDeleteCategory(unsigned int *categoryId*);

Parameters

- ***categoryId*** The web channel category ID.

Returns

None

webchannelDeleteCategoryChannel( )

Removes a channel from a category.

Pseudocode

webchannelDeleteCategoryChannel(unsigned int *ampChannelId*);

Parameters

- ***ampChannelId*** The ID of the web channel to be deleted.

Returns

None

webchannelFindPublicChannels( )

Searches for public channels by name or description or both.

Pseudocode

webchannelFindPublicChannels(
  boolean *searchByTitle*,
  boolean *searchByDescription*,
  string *searchText*
)

Parameters

- ***searchByTitle*** Search for a public channel by name.
- ***searchByDescription*** Search for a public channel by description.
- ***searchText*** Search for a public channel by a search string.

Returns

The information in a `BasicPublicChannelArray` array.

Remarks

To get all channels, pass in: `false`, `false`, `""`.

webchannelGetAllCategories( )

Retrieves a list of all existing public channel categories.

Pseudocode

webchannelGetAllCategories( )

Parameters

- ***None***

Returns

The information in a `CategoryArray` array.

Remarks

To get all channels, pass in: `false, false, ""`

webchannelGetCategoryChannelCount( )

Retrieves the number of channels within each category.

Pseudocode

webchannelGetCategoryChannelCount( );

Parameters

- ***None***

Returns

The information in a `CategoryChannelCountArray` array.

webchannelGetCategoryInfo( )

Retrieves information about a category.

Pseudocode

webchannelGetCategoryInfo(unsigned int *categoryId*);

Parameters

- ***categoryId*** The unique category ID.

Returns

The information in a `Category`structure.

Remarks

To get all channels, pass in: `false, false, ""`

webchannelGetPublicChannelIds( )

Retrieves a list of all public channel Ids.

Pseudocode

webchannelGetPublicChannelIds(
  unsigned int *userId*,
  unsigned int *PublicChannelId*
);

Parameters

- ***userId*** The unique user ID.
- ***PublicChannelId*** The unique public channel ID.

Returns

The information in a `PublicChannelIdArray` array.

webchannelMoveCategoryToTopLevel( )

Makes a sub-category a top-level category.

Pseudocode

webchannelMoveCategoryToTopLevel(unsigned int *categoryId*);

Parameters

- ***categoryId*** The unique category ID.

Returns

None

webchannelSubscribeToPublicChannel( )

Subscribes a user to a public channel.

Pseudocode

```
webchannelSubscribeToPublicChannel(
  unsigned int userId,
  unsigned int PublicChannelId
);
```

Parameters

- ***userId*** The unique user ID.
- ***PublicChannelId*** The unique public channel ID.

Returns

None

webchannelUnsubscribeFromPublicChannel( )

Removes a public channel from the user's subscriptions.

Pseudocode

```
webchannelUnsubscribeFromPublicChannel(
  unsigned int userId,
  unsigned int PublicChannelId
);
```

Parameters

- ***userId*** The unique user ID.
- ***PublicChannelId*** The unique public channel ID.

Returns

None

webchannelUpdateCategory( )

Modifies a web channel category.

Pseudocode

```
webchannelUpdateCategory(
  unsigned int categoryId,
  NewCategory category
);
```

Parameters

- ***categoryId*** The unique category ID.
- ***category*** The name of the category.

Returns

None

Public channel data structures

A public web channel is an URL that you can synchronize to a device. These types of channels are placed in a list which any user can optionally select. The concept is similar to the channels you see on the avantgo.com web site.

BasicPublicChannel

The basic public channel information required to display a list of channels.

Structure

```
{
    unsigned int id;
    string title;
    string description;
    unsigned int categoryId;
    string categoryName;
    unsigned int userCount;
};
```

Fields

- ***id*** Unique ID of this channel.
- ***title*** Display name of channel.
- ***description*** Description of channel.
- ***categoryId*** Category ID (if any).
- ***categoryName*** Name of category (if any).
- ***userCount*** Number of users of this channel.

Remarks

`BasicPublicChannelArray` is an array of `BasicPublicChannel` structures.

Category

Public channels are organized by category, similar to a folder concept. This structure is used when showing a list of categories. Public channel information required to display a list of channels.

Structure

```
{
    unsigned int id;
    unsigned int parent;
    string name;
    string description;
};
```

Fields

- ***id*** Unique ID of this category.
- ***parent*** ID of the parent, if any.
- ***name*** Name of the category.
- ***description*** Description of the category.

Remarks

`CategoryArray` is an array of `Category` structures.

CategoryChannelCount

channels per category, used for display.

Structure

```
{
    unsigned int catId;
    unsigned int count;
};
```

Fields

- ***catId*** Unique ID of this category.
- ***count*** Number of channels in category.

Remarks

`CategoryChannelCountArray` is an array of `CategoryChannelCount` structures.

NewCategory

Use to create/update a category.

Structure

```
{
    unsigned int parent;
    string name;
    string description;
};
```

Fields

- ***parent*** ID of the parent, if any.
- ***name*** Name of the category.
- ***description*** Description of the category.

PublicChannelId

The `agsub` and `amp` channel IDs for a user.

Structure

```
{
    unsigned int agsubId;
    unsigned int ampId;
};
```

Fields

- ***agsubId*** Sub ID.
- ***ampId*** Amp ID.

Remarks

`PublicChannelIdArray` is an array of `PublicChannelId` structures.

Report functions

Use the following generic SQL query function for generating reports.

sqlQuery( )

Executes the SQL statement and returns any results.

Pseudocode

sqlQuery(string *query*);

Parameters

- ***query*** The query string.

Returns

The information in a `Resultset` structure.

Report data structures

Report data structures are the response to executing a generic SQL query. They are used by the administration web application to generate reports.

Resultset

The parameter `rowValues` is a string array of all the values in all the rows. Its size is `numRows x numColumns`. Results are formatted this way to make it easy for clients to read the data.

Structure

```
{
    unsigned int numRows;
    unsigned int numColumns;
    StringArray columnNames;
    StringArray rowValues;
};
```

Fields

- ***numRows*** Number of rows returned.
- ***numColumns*** Number of columns returned.
- ***columnNames*** Column names.
- ***rowValues*** Row values (size == rows x cols).

Remarks

`StringArray` is an array of strings.

M-Business Server configuration functions

Use the following functions to configure your M-Business Server.

configAddCert( )

Adds a certificate for M-Business Server to use in authentication.

Pseudocode

```
configAddCert(
  unsigned int certtype,
  string certFile
);
```

Parameters

- ***certtype*** The certificate type.
- ***certFile*** The name of the certificate file.

Returns

None

configAddLicense( )

Adds a certificate for M-Business Server to use in authentication.

Pseudocode

```
configAddLicense(string licenseVal);
```

Parameters

- ***licenseVal*** The text string comprising the license.

Returns

None

configAllowSelfRegistration( )

Determines whether self-registration is enabled. This function is used on the admin UI login screen to determine if it should display the self registration button. The function checks the self registration preference in the database and it checks the user count against the maximum number of licensed users.

Pseudocode

```
configAllowSelfRegistration( );
```

Parameters

None

Returns

A boolean value.

Remarks

May be called without being logged in. INTERNAL - In future releases, may not be backward compatible or may not be supported at all.

configDelCert( )

Deletes an M-Business Server certificate.

Pseudocode

configDelCert(unsigned int *id*);

Parameters

- ***id*** The certificate ID.

Returns

None

configDelLicense( )

Deletes an M-Business Server license.

Pseudocode

configDelLicense(unsigned int *id*);

Parameters

- ***id*** The certificate ID.

Returns

None

configEnableConduit( )

Turns a conduit on or off system wide.

Pseudocode

configEnableConduit(
  unsigned int *conduitId*,

boolean *enable*
);

Parameters

- ***conduitId*** The unique conduit ID.
- ***enable*** Set to `true`, enables the conduit; set to `false`, disables the conduit.

Returns

None

Remarks

INTERNAL - In future releases, may not be backward compatible or may not be supported at all.

configEnablePersonalChannels( )

Turns on/off personal channels for all users.

Pseudocode

configEnablePersonalChannels(boolean *enable*);

Parameters

- ***enable*** Turn on personal channel conduits for users.

Returns

None

configEnableSelfRegistration( )

Turns on/off self-registration in the web administration application.

Pseudocode

configEnableSelfRegistration(boolean *enable*);

Parameters

- ***enable*** Turn on self registration for users.

Returns

None

Remarks

INTERNAL - In future releases, may not be backward compatible or may not be supported at all.

configGetInfo( )

Retrieves M-Business Server configuration information.

Pseudocode

configGetInfo( );

Parameters

- ***None***

Returns

The information in a `Resultset` structure.

configSetAdminPassword( )

Changes the built-in administrator password.

Pseudocode

```
configSetAdminPassword(
  string oldPassword,
  string newPassword
);
```

Parameters

- ***oldPassword*** The existing administrator password.
- ***newPassword*** The new administrator password.

Returns

None

configSetMinPasswordLength( )

Sets the minimum required password length for all new users.

Pseudocode

configSetMinPasswordLength(unsigned int *length*);

Parameters

- ***length*** The minimum password length for a new user.

Returns

None

configValidateCert( )

Validates an M-Business Server certificate.

Pseudocode

```
configValidateCert(
  unsigned int certtype,
  string certFile
);
```

Parameters

- ***certtype*** The certificate type.
- ***certFile*** The name of the certificate file.

Returns

None

configValidateLicense( )

Validates an M-Business Server certificate.

Pseudocode

```
configValidateLicense(string licenseVal);
```

Parameters

- ***licenseVal*** The text string comprising the license.

Returns

None

M-Business Server configuration data structures

These structures contain general current M-Business Server configuration information. This information is mainly used for server state checking (for example, NT domain integrated or not) and to generate the Status page you see when you first log in to the M-Business Anywhere web administration application as the `admin` user.

ServerConfig

Complete server configuration.

Structure

```
{
   boolean licenseValid;
```



```
    unsigned int licenseType;
    unsigned int minPasswordLength;
    unsigned int totalUsers;
    unsigned int totalGroups;
    unsigned int licenseExpirationFlag;
    long licenseExpirationDate;
    long databaseTime;
    string serverVersion;
    boolean allowPersonalChannels;
    boolean allowSelfRegistration;
    boolean webConduitEnabled;
    boolean xmlConduitEnabled;
    unsigned int serverType;
    string SyncServerPort;
    unsigned int NTDomain;
    string syncsToday;
    string avgSyncTimeToday;
    unsigned int licensedUsers;
};
```

Fields

- ***licenseValid*** Server license valid?
- ***licenseType*** Server license type?
- ***minPasswordLength*** Minimum password length.
- ***totalUsers*** Total number of users.
- ***totalGroups*** Total number of groups.
- ***licenseExpirationFlag*** License flags.
- ***licenseExpirationDate*** License expires date.
- ***databaseTime*** Current time from the DB.
- ***serverVersion*** Version of the soap server.
- ***allowPersonalChannels*** Allow personal channels?
- ***allowSelfRegistration*** Allow self registration?
- ***webConduitEnabled*** Synchronize web channels?
- ***xmlConduitEnabled*** Synchronize database channels?
- ***serverType*** Server type.
- ***SyncServerPort*** Synchronize server port.
- ***NTDomain*** NTDomain/LDAP integrated?
- ***syncsToday*** Number of synchronizations today.
- ***avgSyncTimeToday*** Average synchronization time today.
- ***licensedUsers*** Number of licensed users.

Remarks

`StringArray` is an array of strings. INTERNAL - In future releases, may not be backward compatible or may not be supported at all.

Additional information

Refer to the following resources to obtain additional SOAP information.

- The C# class generated from *avantgoapi.wsdl* is *<M-Business_Home>/samples/csharp/Web References/SoapRef/Reference.cs*.
- Header info:

  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconusingsoapheaders.asp
- SOAP debugging tools:

  http://www.gotdotnet.com/team/tools/Web_svc/

  http://apps.gotdotnet.com/xmltools/WsdlBrowser/

CHAPTER 15

Utilities

Contents

Overview

This section provides application details, including object tag parameters and APIs for using the following M-Business PODS:

- "M-Business Date/Time Picker API" on page 454
- "M-Business List Viewer API" on page 465
- "Scanner" on page 482
- "Signature capture" on page 483

M-Business XML conduit could also be considered as a PODS utility, but is covered in a separate chapter — see "M-Business XML API reference" on page 331.

See "Tools to add special features to a channel" [*M-Business Anywhere Application Developer Guide*], for application details including examples of how the plug-ins appear on both a Microsoft OS device and a Palm OS emulator.

Compatibility between M-Business Client versions and M-Business Server PODS shipped

The utilities documented in this chapter, plus M-Business XML conduit documented in "M-Business XML API reference" on page 331, are supported on mobile devices by platform-specific PODS that ship with M-Business Server. Sometimes a newer version of these PODS is not compatible with an earlier version of M-Business Client. For details on these incompatibilities and how to avoid them, see "M-Business version-specific on-device code files" [*M-Business Anywhere Application Developer Guide*].

Date/Time Picker

The M-Business Date/Time Picker is a MIME player that allows the user to select a date and/or a time. A Date/Time Picker instance is initially displayed as a read-only text field. The user clicks on the text field to expand the Date/Time Picker inline; the user clicks on the text field again to return the Date/Time Picker to its original state.

To embed an instance of the Date/Time Picker in HTML, use the `object` tag and the following application/avantgo-mime-datetimepicker MIME type:

```
<object type="application/avantgo-mime-datetimepicker" width="100"
height="24"> </object>
```


List Viewer

The List Viewer displays the contents of a M-Business XML datastore `agdbset` or of an `ExtendedDBSet`. To embed an instance of the List MIME player in HTML, use the `object` tag and the `application/avantgo-mime-list` MIME type.

M-Business Date/Time Picker API

The M-Business Date/Time Picker is a MIME player that allows the user to select a date and/or a time. A Date/Time Picker instance is initially displayed as a read-only text field. The user taps on the text field to expand the date/time picker in-line; the user taps on the text field again to return the Date/Time Picker to its original state.

Note
The Date/Time Picker currently is only available for use with the M-Business XML POD, which receives data from the M-Business XML conduit. If you use the UltraLite on-device database, you will have to provide comparable functionality.

Caution
If you use the Date/Time Picker, you must make sure that the supporting code file on the device is the correct one for the version of M-Business Client on the device. For more information, see "M-Business version-specific on-device code files" [*M-Business Anywhere Application Developer Guide*].

Specific format specifications for the read-only text field are listed in "Field format specifications" [*M-Business Anywhere Application Developer Guide*].

The rest of this section documents the functions available in the M-Business Date/Time Picker API. For instructions on embedding an instance of the Date/Time Picker in HTML, see "Using the Date/Time Picker" [*M-Business Anywhere Application Developer Guide*].

Summary of M-Business Date/Time Picker API attributes and methods

Description	Attributes and methods
Managing screen display	"refreshScreen( )" on page 459 "setAbbreviatedMonthLabels( )" on page 461 "setAbbreviatedWeekdayLabels( )" on page 462 "setFormat( )" on page 461 "setFullMonthLabels( )" on page 462 "setFullWeekdayLabels( )" on page 463 "setShortWeekdayLabels( )" on page 463

Description	Attributes and methods
Managing date alone	"getDate( )" on page 455 "isDateSet( )" on page 458 "getMonth( )" on page 457 "getYear( )" on page 458 "isNull( )" on page 458
Managing date and time together	"getDateTime( )" on page 455 "getDateTimeString( )" on page 456 "setDateTime( )" on page 460
Managing time alone	"getHours( )" on page 456 "getMinutes( )" on page 457 "isTimeSet( )" on page 459 "setCurrentTime( )" on page 460

getDate( )

Returns the day of the month for the specified date according to local time.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**getDate( )**

C synopsis

AUInt16 **getDate(**DTPickerInstance* *inst***)**

Parameters

- **inst** The MIME instance.

Returns

The day of the month for the specified date according to local time.

getDateTime( )

Returns the date/time value for the instance.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**getDateTime( )**

C synopsis

PODSDate **getDateTime(**DTPickerInstance* *inst***)**

Parameters

♦ ***inst*** The MIME instance.

Returns

An integer value representing the number of seconds since Jan. 1, 1970 00:00:00.

getDateTimeString( )

Returns the date/time string value for the instance.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**getDateTimeString( )**

C synopsis

ADOMString **getDateTimeString(**DTPickerInstance* *inst***)**

Parameters

♦ ***inst*** The MIME instance.

Returns

The string value displayed in the MIME instance.

getHours( )

Returns the hour in the specified date according to local time.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**getHours( )**

C synopsis

AUInt16 **getHours(**DTPickerInstance* *inst***)**

Parameters

- **inst** The MIME instance.

Returns

The hour in the specified time according to local time.

getMinutes( )

Returns the minutes in the specified time according to local time.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**getMinutes( )**

C synopsis

AUInt16 **getMinutes(**DTPickerInstance* *inst***)**

Parameters

- **inst** The MIME instance.

Returns

The minutes in the specified time according to local time.

getMonth( )

Returns the month in the specified date according to the local time.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**getMonth( )**

C synopsis

AUInt16 **getMonth(**DTPickerInstance* *inst***)**

Parameters

- **inst** The MIME instance.

Returns

The month in the specified date according to local time.

getYear( )

Returns the year in the specified date according to the local time.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**getYear( )**

C synopsis

AUInt16 **getYear(**DTPickerInstance* *inst***)**

Parameters

- **inst** The MIME instance.

Returns

The year in the specified date according to local time.

isDateSet( )

Returns if the user has selected a date.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**isDateSet( )**

C synopsis

ABoolean **isDateSet(**DTPickerInstance* *inst***)**

Parameters

- **inst** The MIME instance.

Returns

A boolean value indicating whether the user has selected a date.

isNull( )

Determines whether Date/Time value is null.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**isNull( )**

C synopsis

ABoolean **isNull(**DTPickerInstance* *inst***)**

Parameters

- ***inst*** The MIME instance.

Returns

A boolean value indicating whether the Date/Time value is null.

isTimeSet( )

Returns if the user has selected a time.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**isTimeSet( )**

C synopsis

ABoolean **isTimeSet(**DTPickerInstance* *inst***)**

Parameters

- ***inst*** The MIME instance.

Returns

A boolean value indicating whether the user has selected a time.

refreshScreen( )

Redraws the MIME instance.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**refreshScreen( )**

C synopsis

void **refreshScreen(**DTPickerInstance* *inst***)**

Parameters

- ***inst*** The MIME instance.

Returns

None

setCurrentTime( )

Sets the flag to set current time when a day is selected.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**setCurrentTime(***v***)**

C synopsis

void **setCurrentTime(**
DTPickerInstance* *inst***,**
ABoolean *v*
)

Parameters

- ***inst*** The MIME instance.
- ***v*** A boolean value, true or false.

Returns

None

setDateTime( )

Specifies the date/time value for the instance.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**setDateTime(***dt***)**

C synopsis

void **setDateTime(**
DTPickerInstance* *inst***,**


```
  PODSDate dt
)
```

Parameters

- ***inst*** The MIME instance.
- ***dt*** The number of seconds since Jan. 1, 1970 00:00:00.

Returns

None

setFormat( )

Sets the format specifications for the instance.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**setFormat(***format***)**

C synopsis

```
void setFormat(
  DTPickerInstance* inst,
  ADOMString format
)
```

Parameters

- ***inst*** The MIME instance.
- ***format*** The format specifications for the display. See "Field format specifications" [*M-Business Anywhere Application Developer Guide*], for a list of specification values and their associated descriptions.

Returns

None

setAbbreviatedMonthLabels( )

Sets the labels to be used for abbreviated names of months of year.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**setAbbreviatedMonthLabels(***labels***)**

C synopsis

```
void setAbbreviatedMonthLabels(
  DTPickerInstance* inst,
  ADOMString labels
)
```

Parameters

- ***inst*** The MIME instance.
- ***labels*** Comma separated list of abbreviated month labels.

Returns

None

setAbbreviatedWeekdayLabels( )

Sets the labels to be used for abbreviated names of days of week.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**setAbbreviatedWeekdayLabels(***labels***)**

C synopsis

```
void setAbbreviatedWeekdayLabels(
  DTPickerInstance* inst,
  ADOMString labels
)
```

Parameters

- ***inst*** The MIME instance.
- ***labels*** Comma separated list of abbreviated weekday labels.

Returns

None

setFullMonthLabels( )

Sets the labels to be used for full names of months of year.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**setFullMonthLabels(***labels***)**

C synopsis

void **setFullMonthLabels(**
DTPickerInstance* *inst*,
ADOMString *labels*
)

Parameters

- ***inst*** The MIME instance.
- ***labels*** Comma separated list of full month labels.

Returns

None

setFullWeekdayLabels( )

Sets the labels to be used for full names of days of week.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**setFullWeekdayLabels(***labels***)**

C synopsis

void **setFullWeekdayLabels(**
DTPickerInstance* *inst*,
ADOMString *labels*
)

Parameters

- ***inst*** The MIME instance.
- ***labels*** Comma separated list of full weekday labels.

Returns

None

setShortWeekdayLabels( )

Sets the labels to be used for short names of days of week.

Interface

`mimedtpicker`

JavaScript synopsis

inst.**setShortWeekdayLabels(***labels***)**

C synopsis

```
void setShortWeekdayLabels(
  DTPickerInstance* inst,
  ADOMString labels
)
```

Parameters

- ***inst*** The MIME instance.
- ***labels*** Comma separated list of short weekday labels.

Returns

None

M-Business List Viewer API

The M-Business List Viewer is a MIME player that allows the user to sort and filter the displayed contents of an M-Business XML datastore `agdbset` or of an `ExtendedDBSet`.

Note
The List Viewer currently is only available for use with the M-Business XML POD, which receives data from the M-Business XML conduit. If you use the UltraLite on-device database, you will have to provide comparable functionality.

Caution
If you use the List Viewer, you must make sure that the supporting code file on the device is the correct one for the version of M-Business Client on the device. For more information, see "M-Business version-specific on-device code files" [*M-Business Anywhere Application Developer Guide*].

Specific format specifications for the read-only text field are listed in "Field format specifications" [*M-Business Anywhere Application Developer Guide*].

The rest of this section documents the functions available in the M-Business List Viewer API. For instructions on embedding an instance of the List Viewer in HTML, see "Using the List Viewer" [*M-Business Anywhere Application Developer Guide*].

Summary of M-Business List Viewer API attributes and methods

Description	Attributes and methods
Managing dataset linkages	"getDbSet( )" on page 469 "getSelectionDbSetIndex( )" on page 471 "refreshDbSet( )" on page 473 "setColumnDSName( )" on page 476 "setDbSet( )" on page 479 "setSelectionDbSetIndex( )" on page 480

Description	Attributes and methods
Configuring and formatting columns	“getColumnId( )” on page 466 “getColumnResizeFlag( )” on page 467 “getColumnWidth( )” on page 467 “setColumnAlignment( )” on page 475 “setColumnCount( )” on page 475 “setColumnDSName( )” on page 476 “setColumnFormat( )” on page 476 “setColumnHeaderImage( )” on page 477 “setColumnSortOrder( )” on page 478 “setColumnTitle( )” on page 478 “setColumnWidth( )” on page 479
Managing the screen display	“getColumnId( )” on page 466 “getComponent( )” on page 468 “getScrollPosition( )” on page 471 “getSelectionIndex( )” on page 472 “pageDown( )” on page 472 “pageUp( )” on page 473 “refreshScreen( )” on page 473 “scrollDown( )” on page 474 “scrollUp( )” on page 474 “setScrollPosition( )” on page 480
Accessing onClick event information	“getOnClickEventColumn( )” on page 469 “getOnClickEventRow( )” on page 470 “getOnClickEventTarget( )” on page 470

getColumnId( )

Returns the column index (zero-based) of a location in the Document Coordinate System.

Interface

```
mimelist
```

JavaScript synopsis

inst.**getColumnId(***x*, *y***)**

C synopsis

AInt16 **getColumnId(**
ListInstance* *inst***,**
AInt32 *x***,**
AInt32 *y*
)

Parameters

- ***inst*** The MIME instance.
- ***x*** x position in the Document Coordinate System.
- ***y*** y position in the Document Coordinate System.

Returns

The column index (zero-based), or `-1` if (x, y) doesn't fall within a column.

getColumnResizeFlag( )

Returns the column resize flag since the reset of the dbset.

Interface

`mimelist`

JavaScript synopsis

inst.**getColumnResizeFlag( )**

C synopsis

ABoolean **getColumnResizeFlag(**ListInstance* *inst***)**

Parameters

- ***inst*** The MIME instance.

Returns

The column resize flag since the reset of the dbset.

getColumnWidth( )

Returns the column index (zero-based) of a location in the Document Coordinate System.

Interface

`mimelist`

JavaScript synopsis

inst.**getColumnWidth(***n***)**

C synopsis

AInt16 **getColumnWidth(**
  ListInstance* *inst*,
  AUInt16 *n*
)

Parameters

- ***inst*** The MIME instance.
- ***x*** The column width.

Returns

The column index (zero-based), or `-1` if (x, y) doesn't fall within a column.

getComponent( )

Returns the component at position (*x*, *y*).

Interface

`mimelist`

JavaScript synopsis

inst.**getComponent(***x*, *y***)**

C synopsis

AInt16 **getComponent(**
  ListInstance* *inst*,
  AInt32 *x*,
  AInt32 *y*
)

Parameters

- ***inst*** The MIME instance.
- ***x*** x position in the Document Coordinate System.
- ***y*** y position in the Document Coordinate System.

Returns

0 - TARGET_NONE

1 - TARGET_PREV-PAGE-BTN

2 - TARGET_NEXT_PAGE_BTN

3 - TARGET_HEADER

4 - TARGET_ITEM

5 - TARGET_SCROLL_UP_BTN

6 - TARGET_SCROLL_DOWN_BTN

7 - TARGET_SCROLLBAR_BUBBLE

getDbSet( )

Gets the data source for the MIME instance.

Interface

`mimelist`

JavaScript synopsis

inst.**getDbSet( )**

C synopsis

AGDBSet* **getDbSet(**ListInstance* *inst***)**

Parameters

- ***inst*** The MIME instance.

Returns

The data source for the MIME instance.

getOnClickEventColumn( )

Gets the column number of the onClick event.

Interface

`mimelist`

JavaScript synopsis

inst.**getOnClickEventColumn( )**

C synopsis

AInt16 **getOnClickEventColumn(**ListInstance* *inst***)**

Parameters

- ***inst*** The MIME instance.

Returns

The zero based column number of the onClick event. Returns -1 if not applicable.

getOnClickEventRow( )

Gets the row number of the onClick event.

Interface

`mimelist`

JavaScript synopsis

inst.**getOnClickEventRow( )**

C synopsis

AInt16 **getOnClickEventRow(**ListInstance* *inst***)**

Parameters

- ***inst*** The MIME instance.

Returns

The zero based row number of the onClick event. Returns -1 if not applicable.

getOnClickEventTarget( )

Gets the onClick event target.

Interface

`mimelist`

JavaScript synopsis

inst.**getOnClickEventRow( )**

C synopsis

AInt16 **getOnClickEventRow(**ListInstance* *inst***)**

Parameters

- ***inst*** The MIME instance.

Returns

One of the following values, with the meaning indicated by the name of the associated constant:

- 0 - `TARGET_NONE`
- 1 - `TARGET_PREV_PAGE_BTN`
- 2 - `TARGET_NEXT_PAGE_BTN`
- 3 - `TARGET_HEADER`
- 4 - `TARGET_ITEM`

- 5 - TARGET_SCROLL_UP_BTN
- 6 - TARGET_SCROLL_DOWN_BTN
- 7 - TARGET_SCROLLBAR_BUBBLE
- 8 - TARGET_HEADER_SEP

Remarks

The event target information in the mouse up event handler is cached so that JavaScript code can call this method to determine where the onClick event has happened. This is a workaround for the inability to retrieve event properties in the JavaScript event handler.

getScrollPosition( )

Returns the index of the scroll position (first row of the current page).

Interface

mimelist

JavaScript synopsis

inst.**getScrollPosition( )**

C synopsis

AInt32 **getScrollPosition(**ListInstance* *inst***)**

Parameters

- ***inst*** The MIME instance.

Returns

The index of the first row of the current page.

getSelectionDbSetIndex( )

Returns the database set index of the selected row.

Interface

mimelist

JavaScript synopsis

inst.**getSelectionDbSetIndex( )**

C synopsis

AInt32 **getSelectionDbSetIndex(**ListInstance* *inst***)**

Parameters

- ***inst*** The MIME instance.

Returns

The database set index of the selected row, or `-1` if no row is selected.

getSelectionIndex( )

Returns the offset of the selected row within the currently displayed page. If the first row of the current page is selected, `0` is returned.

Interface

`mimelist`

JavaScript synopsis

inst.**getSelectionIndex( )**

C synopsis

AInt32 **getSelectionIndex(**ListInstance* *inst***)**

Parameters

- ***inst*** The MIME instance.

Returns

The offset of the selected row within the currently displayed page, or `-1` if no row is selected.

pageDown( )

Scrolls the display down one page.

Interface

`mimelist`

JavaScript synopsis

inst.**pageDown( )**

C synopsis

void **pageDown(**ListInstance* *inst***)**

Parameters

- ***inst*** The MIME instance.

Returns

None

pageUp( )

Scrolls the display up one page.

Interface

`mimelist`

JavaScript synopsis

inst.**pageUp( )**

C synopsis

void **pageUp(**ListInstance* *inst***)**

Parameters

- ***inst*** The MIME instance.

Returns

None

refreshDbSet( )

Refreshes the contents due to a change in the underlying database set.

Interface

`mimelist`

JavaScript synopsis

inst.**refreshdbset( )**

C synopsis

void **refreshDbSet(**ListInstance* *inst***)**

Parameters

- ***inst*** The MIME instance.

Returns

None

refreshScreen( )

Redraws the MIME interface.

Interface

`mimelist`

JavaScript synopsis

inst.**refreshScreen( )**

C synopsis

void **refreshScreen(**ListInstance* *inst***)**

Parameters

- ***inst*** The MIME instance.

Returns

None

scrollDown( )

Scrolls the display down one line.

Interface

`mimelist`

JavaScript synopsis

inst.**scrollDown( )**

C synopsis

void **scrollDown(**ListInstance* *inst***)**

Parameters

- ***inst*** The MIME instance.

Returns

None

scrollUp( )

Scrolls the display up one line.

Interface

`mimelist`

JavaScript synopsis

inst.**scrollUp( )**

C synopsis

void **scrollUp(**ListInstance* *inst***)**

Parameters

- ***inst*** The MIME instance.

Returns

None

setColumnAlignment( )

Sets the alignment for a column.

Interface

`mimelist`

JavaScript synopsis

inst.**setColumnAlignment(***n*, alignment**)**

C synopsis

void **setColumnAlignment(**
ListInstance* *inst***,**
AUInt *n***,**
AUInt16 *alignment*
)

Parameters

- ***inst*** The MIME instance.
- ***n*** The zero-based column index.
- ***alignment*** The alignment value. Possible values are:

  `0` - ALIGN_LEFT

  `1` - ALIGN_RIGHT

  `2` - ALIGN_CENTER

Returns

None

setColumnCount( )

Sets the number of columns in the MIME instance

Interface

`mimelist`

JavaScript synopsis

inst.**setColumnCount(***count***)**

C synopsis

void **setColumnCount(**
ListInstance* *inst*,
AUInt16 *count*
)

Parameters

- ***inst*** The MIME instance.
- ***count*** The number of columns.

Returns

None

setColumnDSName( )

Sets the data source column name for a column.

Interface

`mimelist`

JavaScript synopsis

inst.**setColumnDSName(***n*, *dataSourceColName***)**

C synopsis

void **setColumnDSName(**
ListInstance* *inst*,
AUInt16 *n*,
ADOMString *dataSourceColName*
)

Parameters

- ***inst*** The MIME instance.
- ***n*** The zero-based column index.
- ***dataSourceColName*** The data source column name.

Returns

None

setColumnFormat( )

Sets the format for a column.

Interface

`mimelist`

JavaScript synopsis

inst.**setColumnFormat(***n*, *format***)**

C synopsis

void **setColumnFormat(**
Instance* *inst*,
AUInt16 *n*,
ADOMString *format*
)

Parameters

- ***inst*** The MIME instance.
- ***n*** The zero-based column index.
- ***format*** The format specification.

Returns

None

setColumnHeaderImage( )

Sets the default header image for a column.

Interface

`mimelist`

JavaScript synopsis

inst.**setColumnHeaderImage(***n*, *img***)**

C synopsis

void **setColumnHeaderImage(**
ListInstance* *inst*,
AUInt16 *n*,
AUInt16 *img*
)

Parameters

- ***inst*** The MIME instance.
- ***n*** The zero-based column index.
- ***img*** The image value. Possible values are:

  `0` - IMAGE_NONE

  `1` - IMAGE_LINE

Returns

None

setColumnSortOrder( )

Sets the sort order of a column.

Interface

`mimelist`

JavaScript synopsis

inst.**setColumnSortOrder(***colId*, *ascending***)**

C synopsis

void **setColumnSortOrder(**
ListInstance* *inst*,
AInt16 *colId*,
ABoolean *ascending*
)

Parameters

- ***inst*** The MIME instance.
- ***colId*** The column index.
- **ascending** True for ascending sorting order; false for descending sort order.

Returns

None

setColumnTitle( )

Sets the title for a column.

Interface

`mimelist`

JavaScript synopsis

inst.**setColumnTitle(***n*, *title***)**

C synopsis

void **setColumnTitle(**
ListInstance* *inst*,
AUInt16 *n*,
ADOMString *title*
)

Parameters

- ***inst*** The MIME instance.
- ***n*** The zero-based column number.
- ***title*** The title for the column.

Returns

None

setColumnWidth( )

Sets the suggested width for a column.

Interface

`mimelist`

JavaScript synopsis

inst.**setColumnWidth(***n*, *width***)**

C synopsis

void **setColumnWidth(**
ListInstance* *inst*,
AUInt16 *n*,
AUInt16 *width*
)

Parameters

- ***inst*** The MIME instance.
- ***n*** The zero-based column index.
- ***width*** The suggested column width.

Returns

None

setDbSet( )

Sets the data source for the MIME instance.

Interface

`mimelist`

JavaScript synopsis

inst.**setDbSet(***dbset***)**

C synopsis

```
void setDbSet(
  ListInstance* inst,
  AGDBSet* dbset
)
```

Parameters

- ***inst*** The MIME instance.
- **dbset** The M-Business XML datastore set.

Returns

None

Remarks

Call `list.setDbSet(null)` after you close the database to inform `mimelist` that the database handle is no longer valid from that point on.

setScrollPosition( )

Scrolls the MIME instance to the given scroll position.

Interface

`mimelist`

JavaScript synopsis

inst.**setScrollPosition(***scrollPosition***)**

C synopsis

```
void setScrollPosition(
  ListInstance* inst,
  AInt32 scrollposition
)
```

Parameters

- ***inst*** The MIME instance.
- ***scrollPosition*** The index of the scroll position.

Returns

None

setSelectionDbSetIndex( )

Makes a database row the current selection.

Interface

`mimelist`

JavaScript synopsis

inst.**setSelectionDbSetIndex(***dbsetIndex***)**

C synopsis

void **setSelectionDbSetIndex(**
ListInstance* *inst***,**
AInt32 *dbsetIndex*
)

Parameters

- ***inst*** The MIME instance.
- **dbsetIndex** The index of a row in the underlying database set.

Returns

None

Scanner

M-Business Anywhere provides a `PODSSymbolScanner` interface that implements a Symbol Technologies API for both Palm OS and WinCE platforms.

The `PODSSymbolScanner` object provides the API for working with a Symbol Technologies bar-code scanner device. Your POD might allow users to update information in a POD-specific database by scanning in that new information. The `PODSSymbolScanner` object provides a way to do this. Refer to "PODSSymbolScanner calls" on page 308 for API details.

Note
You must perform a binary download of the `scanner.dll` file into the `/pods` directory for WinCE devices.

You access the `PODSSymbolScanner` object by using the `PODSWindow` method "createObject( )" on page 85. Because the `PODSWindow` object creates the `PODSSymbolScanner` object, you do not have to free the memory associated with it.

The `PODSSymbolScanner` object's API is essentially a wrapper for the Scan Manager API provided by Symbol Technologies. Refer to the Symbol Technologies documentation for complete descriptions of how to use the Scan Manager.

Signature capture

M-Business Anywhere provides an in-line scribble widget that allows signature capture on the device at the scribble label level. There is no large dialog that pops up, so the signer of the document is clearly signing the agreed-upon document.

For information on implementing signature capture in an HTML page, see "Capturing signatures with an in-line scribble widget" [*M-Business Anywhere Application Developer Guide*].

For reference material on the API calls that are involved, see "PODSSymbolScanner object" on page 308.

Part III. Sample Implementations

Appendix A

PODS code samples

Contents

Downloading and working with the PODS sample files

The sample files included in this section illustrate some of the most common application features implemented through PODS.

All but one of these sample files are contained in a file called *pods.zip*. To download the *pods.zip* file:

♦ To download the pods.zip file

1. Navigate to the URL below.

   http://www.ianywhere.com/developer/product_manuals/mbusiness_anywhere/index.html

2. For the latest version listed at the top, click the link for either **View HTML** or **View PDF**.
3. Under the **Developer** heading, click the **PODS API resources...** link.
4. Unzip all of the files, preserving the directory information.

   Check **Use folder names** in the WinZip Extract dialog.

   Place the files in any convenient directory (**Extract to** in the WinZip Extract dialog). A *pods* directory and various subdirectories will be extracted below the directory you specify in the "Extract to" box in the WinZip Extract dialog.

To work with a sample file, navigate to the sample file's folder below the *pods* directory, then go into the folder for your target platform. In that folder, open the project file for the sample file and platform.

Caution
These sample files are intended to serve as samples for writing your own code. They may not work as is, without some modification.

For instructions on writing your own PODS source files, see "Writing C Code for PODS" on page 13.

Pod sample: submitting forms

Below is a listing of the *pod.c* sample code file. For instructions on obtaining the source for this sample POD, see "Downloading and working with the PODS sample files" on page 488.

```
/* Copyright 2004 iAnywhere Solutions, Inc. All rights reserved.
 *
 * This sample source code is provided to aid in
 * creation of PODS modules for iAnywhere Solutions software.
 *
 * pod.c
 *
*/

#include "pods.h"
#include "podsstartup.h"

#include "pod.h"
#include "documentsrc.h"

#if 0
/***** getVersion *****/

// This function is special, because ANY PODSObject
// built from a given set of PODS headers must
// return the version constant from that set of
// headers. So we can just make one copy of this
// function and put it into the vtable of any
// object, including the PODSPod itself, that
// we create

PODSUInt16  getVersion(PODSPod *self)
{
   return PODS_VERSION;
}
#endif

PODSString FormSubmitPodGetPodDescription(PODSPod *pod);
PODSString FormSubmitPodGetPodDescription(PODSPod *pod)
{
return "FormSubmit Sample Pod";
}

PODSString FormSubmitPodGetPodVersion(PODSPod *pod);
PODSString FormSubmitPodGetPodVersion(PODSPod *pod)
{
   return "v 1.0";
}

void FormSubmitPodDestroy(PODSPod *pod);
void FormSubmitPodDestroy(PODSPod *pod)
{
   FormSubmitPod *self = (FormSubmitPod *)pod;

   free(self->podsPod.vtable);
   free(self);
}

/***** PODSPodNew *****/

// This is the only entry point to your pod, which
// will be called as soon as the pod is loaded (and
```

```
// after M-Business Client is fully initialized).
// The PODSPod object returned by PODSPodNew is the
// embodiment of your pod, and its Destroy method
// will be invoked just before your pod is unloaded
// (and before any part of M-Business Client is
// shut down).

PODSPod *PODSPodNew(PODSAvantGo *avantgo)
{
   FormSubmitPod *self = (FormSubmitPod *)malloc(sizeof(FormSubmitPod));
   PODSDocumentMgr *docMgr = PODSgetDocumentMgr(avantgo);
   PODSDocumentSrc *docSrc = (PODSDocumentSrc *)DocumentSrcNew(docMgr);

   self->podsPod.avantgo = avantgo;
   self->podsPod.vtable = (PODSPodVTable *)calloc(1, sizeof(PODSPodVTable));

#if 0
   self->podsPod.vtable->m_getVersion = getVersion;
#endif
   self->podsPod.vtable->m_getPodDescription =
FormSubmitPodGetPodDescription;
   self->podsPod.vtable->m_getPodVersion = FormSubmitPodGetPodVersion;
   self->podsPod.vtable->m_destroy = FormSubmitPodDestroy;

PODSregisterDocumentSrc(docMgr, docSrc);

return (PODSPod *)self;
}
```

DocumentSrc sample: vending documents

Below is a listing of the *DocumentSrc.c* sample code file. For instructions on obtaining the source for this sample POD, see "Downloading and working with the PODS sample files" on page 488.

```
/*
 * Copyright 2004 iAnwhere Solutions, Inc. All rights reserved.
 *
 * documentsrc.c
 *
 */

#include "pods.h"
#include "podsstartup.h"
#include "documentsrc.h"

// This is the function that is invoked when the
// document manager is searching for a document. If
// the url matches our SAMPLE_URL, we generate a
// document and return it to the document manager.
static PODSDocument
          *DocumentSrcDocumentForUrl(PODSDocumentSrc *podsDocSrc, PODSString
url, PODSBoolean *handled)
{
   DocumentSrc *self = (DocumentSrc *)podsDocSrc;

   if (0 == stricmp(url, SAMPLE_URL)) {
      PODSDocument *doc = PODScreateDocument(self->documentMgr, SAMPLE_URL,
PODS_HTML_TYPE);
      ADOMHTMLDocument *dom = PODSgetDom(doc);
      ADOMElement *body;
      ADOMText *node;

      ADOMsetTitle(dom, (ADOMString)"Hello World!");

      body = ADOMcreateElement(dom, (ADOMString)"body");
      ADOMappendChild(dom, (ADOMNode *)body);

      node = ADOMcreateTextNode(dom, (ADOMString)"Hello World!");
      ADOMappendChild(body, (ADOMNode *)node);

      if (handled)
         *handled = PODS_TRUE;

      return doc;
   }

   return NULL;
}

static void DocumentSrcDestroy(PODSDocumentSrc *podsDocSrc)
{
   DocumentSrc *self = (DocumentSrc *)podsDocSrc;

   free(self->vtable);
   free(self);
}

// Create our document source and fill in vtable
// entries for documentForUrl and destroy
```

```
   DocumentSrc *DocumentSrcNew(PODSDocumentMgr *documentMgr)
{
   DocumentSrc *self = (DocumentSrc *)calloc(1, sizeof(DocumentSrc));

   self->vtable = (PODSDocumentSrcVTable *)calloc(1,
sizeof(PODSDocumentSrcVTable));

   PODS_SET_METHOD(self->vtable, documentForUrl, DocumentSrcDocumentForUrl);
   PODS_SET_METHOD(self->vtable, destroy, DocumentSrcDestroy);

   self->documentMgr = documentMgr;

   return self;
}
```

ObjectSrc sample: vending objects to JavaScript

Below is a listing of the *ObjectSrc.c* sample code file. For instructions on obtaining the source for this sample POD, see "Downloading and working with the PODS sample files" on page 488.

```
/*
 * Copyright 2004 iAnwhere Solutions, Inc. All rights reserved.
 *
 * objectsrc.c
 *
 */

#include "pods.h"
#include "podsstartup.h"
#include "objectsrc.h"

// This is the function that it invoked when the
// object manager is searching for an object. If
// the name matches our SAMPLE_NAME, we return our
// object (creating it if necessary) and return it
// to the object manager.
static PODSObject
           *ObjectSrcObjectForName(PODSObjectSrc
           *podsObjSrc, PODSString name)
{
   ObjectSrc *self = (ObjectSrc *)podsObjSrc;

   if (0 == strcmp(name, SAMPLE_NAME)) {
      if (!self->sampleObject)
         self->sampleObject = SampleObjectNew();

      return (PODSObject *)self->sampleObject;
   }

   return NULL;
}

static void ObjectSrcDestroy(PODSObjectSrc
            *podsObjSrc)
{
   ObjectSrc *self = (ObjectSrc *)podsObjSrc;

   if (self->sampleObject)
      PODSdestroy((PODSObject *)self->sampleObject);
   free(self->vtable);
   free(self);
}

// Create our object source and fill in vtable
// entries for objectForName and destroy
// ObjectSrc *ObjectSrcNew(PODSObjectMgr *objectMgr)
{
   ObjectSrc *self = (ObjectSrc *)calloc(1, sizeof(ObjectSrc));

   self->vtable = (PODSObjectSrcVTable *)calloc(1,
sizeof(PODSObjectSrcVTable));

   PODS_SET_METHOD(self->vtable, objectForName, ObjectSrcObjectForName);
   PODS_SET_METHOD(self->vtable, destroy, ObjectSrcDestroy);
```

```
        self->objectMgr = objectMgr;
        return self;
    }
```


Forms sample: resetting channels

```
<html>

<form name=foo>

<input type=button
           onClick="javascript:avantgo.resetChannels();" value="Click me to
eliminate
           all WebToGo content from this server">

/form

</html>
```

When you click the button, it goes to the "You have successfully installed the M-Business Client software, but you do not have any channels" message window.

Programmatically exiting M-Business Client

This sample code is not currently included in the *pods.zip* file referenced in "Downloading and working with the PODS sample files" on page 488. This sample shows how you can write code that exits M-Business Client.

```
// vtable
void (*m_exit)(PODSObject*);

// new method
PODS_SET_METHOD(self->vtable, exit, MenuObjectExit);

// GetMethod
if (0 == strcmp(methodName, "exit")) {
    *methodSignature = "";
    return (PODSMethod)MenuObjectExit;
}

// method
static void MenuObjectExit(PODSObject *podsObj)
{
    MenuObjectDestroy( podsObj );
    PostQuitMessage( 0 );
}
```


Appendix B

SOAP sample client files

Contents

Overview of SOAP sample client files

The M-Business Server installer places files for sample SOAP clients written in C# and Java in branches directly under the *<M-Business_Home>/samples/* directory.

You can use these files to set up and test communication between the M-Business SOAP API and M-Business SOAP Server. You execute the SOAP sample client from a command prompt and it provides feedback to the console on success or failure of each API call. The SOAP sample client first adds information to the M-Business Anywhere database, then deletes it. After the SOAP sample client has completed its run, the M-Business Anywhere database is unchanged from its initial state.

Note
All functions are available to M-Business Server's admin user account. Other users may call a subset of these functions. If you see a "permission denied" error, you must be admin to call that function.

C# SOAP sample client

Building and running the C# sample SOAP client

Some of the details in the steps in the *ReadMe.txt* file for C# are outdated. The procedure below has file paths that match those that are installed with M-Business Server version 6.7.

Note
The <*M-Business_Home*> directory referred to in the steps below is the M-Business home directory that is created under the location that you specify during installation. For more information, see "The M-Business home directory" [*M-Business Anywhere Administrator Guide*].

♦ To build and run the C# sample SOAP client

1. Make sure M-Business SOAP Server is running on the default port 8093, and vending WSDL on the default port 8094.
2. From your ***agsoap.conf*** file, copy the **`AuthKey`** into line 68 in ***Main.cs***.
3. Save a copy of the <***M-Business_Home*>*/samples/csharp/Web References/SoapRef/Reference.cs*** file somewhere, in case Visual Studio overwrites it.

   This is the file that Visual Studio generates for you. The version installed with M-Business Server has been modified so that it will interact with standard SOAP web services, like those in the M-Business SOAP API.
4. Open the <***M-Business_Home*>*/samples/csharp/console.sln*** file in Visual Studio .NET.
5. In Visual Studio .NET, choose **Build»Build Solution**.

   The SOAP client built is <*M-Business_Home*>*/samples/csharp/bin/Debug/console.exe*
6. Copy <***M-Business_Home*>*/bin/soap_crypt.dll*** to <***M-Business_Home*>*/samples/csharp/bin/Debug***.
7. From a command prompt, change directories to the directory above:

   `cd <M-Business_Home>/samples/csharp/bin/Debug`
8. From the same command prompt, enter the following commands — replace <*password*> with your current admin user password.:

   `console admin "<password>" "select now()"`

   `console admin "<password>"`

   This will run all the API tests.

Note
Currently, the SOAP message that is generated by .NET is incorrect. The XML SOAP header is wrong. For a description of the problem, see http://www.dotnet247.com/247reference/msgs/25/128422.aspx.
To work around this .NET bug, we have edited a generated file (*Reference.cs*).
For information on the header reference code, see http://soapinterop.java.sun.com/soapbuilders/r4/soapfault.wsdl.
For header information, see http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconusingsoapheaders.asp.

C# SOAP sample client files

Main.cs

The *Main.cs* file logs in to M-Business SOAP Server and sets up an environment from which calls can be made to the M-Business SOAP API.

AsemblyInfo.cs

The *AsemblyInfo.cs* file sets up the assembly for the Microsoft .NET framework that Visual Studio .NET requires.

Hook.cs

The *Hook.cs* file sets up helper classes to properly format SOAP headers. For an explanation of why this is necessary, see comments in the file.

TestAPI.cs

The *TestAPI.cs* sets up some more helper functions, but most of the sample code in this file consists of calls to the M-Business SOAP API, with explanatory comments. The following is the first call, creating a new user:

```
// group user create
NewUser nu = new NewUser();
nu.userName = "CS_UnitTestGroupUser" + System.DateTime.Now.Ticks;
nu.firstName = "UnitTest";
nu.lastName = "User";
nu.password = "j";
nu.comment = "exec";
m_groupUserId = m_api.userCreate(nu);
if(m_groupUserId == 0)
{
    Console.WriteLine("Failed to create group user");
    return 1;
}
Console.Write( "." );
```


Java SOAP sample client

Compiling and running the Java sample SOAP client

This client uses the spheon-jsoap library (version 0.5.0), which can be obtained from http://sourceforge.net/projects/spheon-jsoap/

This sample contains the parts of the library you need. See the link above if you wish to obtain the jsoap documentation.

This sample is built using the JDK that is shipped with M-Business Server version 6.7. The instructions below are specific for a Windows system.

Note
The <*M-Business_Home*> directory referred to in the steps below is the M-Business home directory that is created under the location that you specify during installation. For more information, see "The M-Business home directory" [*M-Business Anywhere Administrator Guide*].

♦ To compile and run the Java sample SOAP client

1. Make sure M-Business SOAP Server is running on the default port 8093, and vending WSDL on the default port 8094.
2. In the <***M-Business_Home*>*/samples/java/*** directory, copy the appropriate file below to overwrite ***Makefile***.
   - On Windows, copy ***Makefile.win32***.
   - On Unix, copy ***Makefile.unix***.
3. Open a command prompt and change directories to the location of the *Main.java* file:

   ```
   cd <M-Business_Home>/samples/java
   ```
4. At the same command prompt, enter the following command to compile the Java sample SOAP client:

   ```
   make
   ```

   Now you are ready to run tests against M-Business SOAP Server.
5. Enter the following command:

   ```
   make test
   ```

   This executes the following command to test the M-Business SOAP Server connection:

   ```
   ../../jdk/bin/java -classpath "lib/spheon-jsoap.jar:lib/spheon-
   jsoap-tools.jar:lib/jdom.jar:lib/xalan.jar:lib/xml-apis.jar:lib/
   xercesImpl.jar:." Main http://localhost:8093/agsoap admin "" "select
   now()"
   ```


Java sample SOAP client files

Main.java

The *Main.java* file logs in to M-Business SOAP Server and sets up an environment from which calls can be made to the M-Business SOAP API.

Test files

The Java files that actually make the test calls to M-Business SOAP Server are located in the *<M-Business_Home>/samples/java/tests/* directory, The file names indicate the groups of functions that each file tests:

- *TestConfigApi.java*
- *TestGroupApi.java*
- *TestUserApi.java*
- *TestWebChannelApi.java*

The following is the first call from *TestConfigApi.java*, logging in to M-Business SOAP Server:

```
// login to a session
try {
    // NOTE: Make sure to set this to the exact same AuthKey string
    //       that is in the ../conf/agsoap.conf file.
    //       (otherwise you will keep seeing login failures because
    //        the server won't be able to decrypt the password)
    String AuthKey = "your.authkey.here";

    // Encrypt the password using the Blowfish
    BlowfishCrypt crypt = new BlowfishCrypt();
    crypt.encrypt( AuthKey, AuthKey.length(),
                   password, password.length() );
    byte[] dest = crypt.getDest();
    String b64password =  base64.encode( dest );

    // actually login as a user now
    RPCCall call = new RPCCall( /* SoapConfig */ sc,
                                /* namespace  */ "urn:AvantgoWebAPI",
                                /* method     */ "loginUser" );

    // set parameters
    call.setParameter( /* name  */ "userName",
                       /* value */ userName,
                       /* type  */ String.class );
    call.setParameter( /* name  */ "b64password",
                       /* value */ b64password,
                       /* type  */ String.class );
```


Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

