BadgeBuilder® XML RPC

Specification

By Nesc, Inc.

This manual was produced using *ComponentOne Doc-To-Help*.™

Contents

Overview

Introduction

Starting with version 4.50 of BadgeBuilder® a new interface is available to retrieve information from and send commands to BadgeBuilder®.

The interface uses XML (eXtended Markup Language) to communication via an HTTP protocol connection, and any client capable of performing XML communication can operate with BadgeBuilder®. The interface is called XML RPC, where RPC stands for Remote Procedure Call.

The interface allows a client program to take over operation of BadgeBuilder® and provides over 60 functions for controlling BadgeBuilder® and gaining access to its data.

Note for TeamTracer® Users

Whereever the word **BadgeBuilder®** appears in this manual the text also applies to **TeamTracer®** unless specifically noted otherwise.

For TeamTracer® the initialization file is called **bbtt.ini**, so whereever **bb32.ini** is referenced in this document, read **bbtt.ini** for TeamTracer®.

HTTP Access

When the BadgeBuilder® XML interface is enabled, it becomes an HTTP server listening on a user-definable port. Assuming a firewall does not prevent it, an XML client can communication with BadgeBuilder® from anywhere on the Intranet or Internet.

Developer's Package

The developer's package includes a Windows™ DLL (Dynamically Linked Library) which implements an XML client. The library is called **BBXMLClientDLL.dll** and two library files, **BBXMLClientDLL.lib** for the Borland™ compiler and **BBXMLClientDLLms.lib** for the Microsoft™ compiler are also provided.

The package also includes two header files defining interface calls and other useful definitions. These files are called **BBXMLClientDLLInterface.h** and **command.h**. These must be included in any C++ program intended to implement an XML RPC client.

Also included is the source for a Borland® Project which creates a test program using the BadgeBuilder® XML RPC client DLL. This program is intended to help the developer understand the XML RPC interface.

Functional Description

Interface Capabilities

The XML RPC interface can be accessed via any XML client by posting XML to the BadgeBuilder® server and interpreting the results. In addition, a Windows® DLL has been provided which implements the XML client and can be linked into Microsoft or Borland applications.

The interface provides a rich set of calls for extracting information from BadgeBuilder® and for commanding BadgeBuilder® to perform functions.

As a result, badging can be added to an end user application quickly and easily and the result is a capable, integrated operation.

XML Access Key

Each XML communication with the BadgeBuilder® XML server must include an access key. BadgeBuilder® starts off with a default key but this can be changed via bb32.ini settings.

The XML client DLL has the default key built into it and also provides a call to change it to match any custom key defined in bb32.ini.

When the server and client are on the same computer the key serves little useful purpose. But when the client is remote from the server it provides a degree of control over who has access to the XML interface.

Setting Up BadgeBuilder

BB32.INI Settings

Unless settings are entered into the **bb32.ini** file resident where BadgeBuilder® was installed, the XML RPC interface is disabled.

To enable the interface place the following lines in the **[status]** section of the file. (This file may be read-only. If so, select its properties and remove the read-only check mark.)

```
XMLPort=8080
XMLAccessKey=Magic Key
```

If the XMLPort value is missing, or set to 0, the XML RPC interface will be disabled. Choose any unused port to enable to interface. A popular choice is 8080. (To discover which ports are currently in use run the **netstat –an** command from a Windows™ XP or 2000 command prompt window.)

The XMLAccessKey value can be any string of characters you wish.

If you plan to use the supplied client DLL then there is no needed to set the XMLAccessKey value. But if you do set it, then you must also tell the DLL what the key is.

If you plan to write your own XML RPC client then you must set the desired key in bb32.ini and use it in your XML communications.

For either entry, do **not** put spaces around the equals (=) sign.

BB32.exe Command Line Switches

BadgeBuilder® can be started with various command line switches which determine its initial running status. These commands are listed here.

/SERVER

When present, BadgeBuilder® will start up immediately in Server Mode, as long as a valid security key (Dongle) is attached.

/MINITOOLBAR

When present, BadgeBuilder® will start up with the standard mini-toolbar enabled. Usually this command is only used in conjunction with the /Server switch.

Note that BadgeBuilder® cannot be shut down by the user when the mini-toolbar is enabled. It must be closed using the XML Exit function.

For more information on the mini-toolbar see ServerMode.

/SHOW=<mode>

Causes BadgeBuilder® to be started in the given visual mode. Possibilities for <mode> are HIDE or MINIMIZE. If <mode> is omitted or invalid, BadgeBuilder® will start up in normal mode.

Remember when using the HIDE mode that the main BadgeBuilder® window will never be seen by the user, so BadgeBuilder® must be closed using the XML Exit function.

/STARTDIRECTORY=<directory>

If set to a valid path, this will be the directory BadgeBuilder® will look in for .IDC files as a default. No spaces are allowed in the command and if <directory> contains spaces then it must be double quoted. For example:

/STARTDIRECTORY="c:\program files\my badges"

/LOG

If present, logging of operations will start up active.

DLL Interface

Introduction

This section provides full details of the calls made available by the client DLL. A basic knowledge of programming is expected, and the target language is C.

Transferring Data

Many calls require data to be passed to the server. Others return complex information such as the contents of a badge record. This is accomplished using a structure defined in the **commands.h** header file. This is the definition:

```
typedef XMLRECORDDATA *
        {
        int   fieldCount;
        char **fieldName;
```

```
    char **fieldData;
}XMLRECORDDATA, *lpXMLRECORDDATA;
```

The fieldCount element contains the number of parameters being passed in the structure. The fieldName element is an array of pointers to character strings containing the name of the parameter. Similarly, the fieldData element contains the value of the parameter.

It is always the responsibility of the creator of the structure to release any memory it may have allocated for the various structure elements.

In the description of the calls which follow, the contents of the structure are represented by name=value pairs. For example, if the definition was:

FirstName=George
LastName=Smith

Then fieldCount would be set to 2, and fieldName would point to an array of 2 pointers to character strings, the first containing **FirstName** and the second containing **LastName**. Similarly, fieldData would point to an array of 2 pointers to character strings, the first containing **George** and the second containing **Smith**.

When an input field is a Boolean value then **1**, **on**, **yes**, and **true** are considered equal and any other value is considered **false**. These are not case-sensitive.

Informational Calls

LaunchBB

Prototype: bool LaunchBB(char *idcfile, int showState)

Description: Assuming BadgeBuilder® has been correctly installed on the machine where the XML client is running, this will launch it and open the given IDC file.

This will only launch BadgeBuilder® on the machine where the XML client is running.

Input Data Format:
idcfile: Character pointer to a fully qualified BadgeBuilder® IDC file.
showState: visible state of BadgeBuilder® on start-up, should be one of the Windows™ API constants SW_NORMAL, SW_MINIMIZED, SW_HIDE, etc.

Returned Data: True if successful. If unsuccessful, use the GetErrorCode call to obtain the error code from the Windows™ ShellExecuteEx API call. The GetErrorText can be used to get the text of the error message.

LaunchTT

Prototype: bool LaunchTT(char *idcfile, int showState)

Description: Assuming TeamTracer® has been correctly installed on the machine where the XML client is running, this will launch it and open the given IDC file.

This will only launch TeamTracer® on the machine where the XML client is running.

Input Data Format:
idcfile: Character pointer to a fully qualified TeamTracer® IDC file.

showState: visible state of TeamTracer® on start-up, should be one of the Windows™ API constants SW_NORMAL, SW_MINIMIZED, SW_HIDE, etc.

Returned Data: True if successful. If unsuccessful, use the GetErrorCode call to obtain the error code from the Windows™ ShellExecuteEx API call. The GetErrorText can be used to get the text of the error message.

SetXMLPort

Prototype: bool SetXMLPort(int port)

Description: Set the port which BadgeBuilder® is listening to. The default for the XML client DLL is 8080, but this may have been changed for the BadgeBuilder® server in the bb32.ini file.

Input Data Format: Integer value for the server port to access

Returned Data: Always returns **True**.

SetHostName

Prototype: bool SetHostName(char *name)

Description: Set the host name where BadgeBuilder® is running. If this is the same machine as the XML client, then set this to **localhost.** Alternatively an Internet IP can be set. A normal URL (such as mydomain.com) can also be used as long as it is known to the DNS server configured for the machine where the XML client is running.

Input Data Format: character pointer to the name of the host where BadgeBuilder® is running

Returned Data: Always returns True.

SetXMLAccessKey

Prototype: bool SetXMLAccessKey(char *name)

Description: Set the authorization key when communication with the BadgeBuilder® server. If the BadgeBuilder® server authorization key has not been changed by an entry in bb32.ini, then there is no need to set this here. If bb32.ini does set a new key then this call must be made to make the client match it.

Input Data Format: character pointer to the new access key.

Returned Data: Always returns True.

CloseXML

Prototype: bool CloseXML(void)

Description: Close the XML link to the BadgeBuilder® server. There is no reason to call this as the link is automatically closed when the program using the DLL is closed.

Returned Data: Always returns True.

EnableXMLAccess

Prototype: bool EnableXMLAccess(bool flag)

Description: Enables and disables access to the actual XML sent and received over the link. This is initialized to disabled as it is not needed for normal operation and can have a performance impact if enabled unnecessarily. Until this is enabled, the calls **GetSentXML** and **GetReceivedXML** will not return anything useful.

Input Data Format: Boolean True or False

Returned Data: Always returns True.

GetSentXML

Prototype: char * GetSentXML(void)

Description: If EnableXMLAccess is set to True, this will return the XML code sent to the server after the call has been made.

Returned Data: character pointer to XML code

GetReceivedXML

Prototype: char * GetReceivedXML (void)

Description: If EnableXMLAccess is set to True, this will return the XML code received from the server after the call has been made.

Returned Data: character pointer to XML code

GetErrorCode

Prototype: int GetErrorCode(void)

Description: Returns the error code of the last call executed. For pending commands this is just whether or not the call has been made pending or not. Call GetLastCommandError to find out the result of the pending command once it has been executed.

Returned Data: integer containing the error code.

GetErrorText

Prototype: char * GetErrorText(void)

Description: Returns the error text of the last call executed. For pending commands this is just whether or not the call has been made pending or not. Call GetLastCommandError to find out the result of the pending command once it has been executed and pass it to GetErrorMessage to get the text.

Note that this call will not return useful information for the LaunchBB call.

Returned Data: character string containing the last call's error text.

GetMethods

Prototype: XMLRECORDDATA * GetMethods(void)

Description: Returns a structure containing the names of all the XML calls supported by the BadgeBuilder® server. Note that calls starting with **system.** are provided for introspection capability, to support this call and **GetMethodProtoType** and **GetMethodHelp**.

Example Returned Data:
Element1=GetErrorMessage
Element2= IsBusy
Element3= GetLastCommandError
.
.
etc.

GetMethodProtoType

Prototype: XMLRECORDDATA * GetMethodProtoType(char *method)

Description: Returns a string containing the prototype for the given method.

Input Data Format: character pointer to the name of the method

Example Returned Data:
Element1 = char * GetErrorMessage(struct data)

GetMethodHelp

Prototype: char * GetMethodHelp(char *method)

Description: Returns technical information about the call defined by the method parameter.

Input Data Format: character pointer to the name of the method

Returned Data: character string containing the help information

GetErrorMessage

Prototype: char * GetErrorMessage(RecordData *data)

Description: Returns the error message string associated with the error code number supplied.

Input Data Format:
ErrorCode = <code number>

Returned Data: If the error code supplied is valid, the returned character pointer points to a string containing the message text associated with the error code. If the error code is not valid the return pointer is set to NULL.

IsBusy

Prototype: bool IsBusy(void)

Description: Returns **True** if a command is already pending which requires BadgeBuilder® to be in the main loop, and **False** otherwise. Before issuing a command which uses the pending mechanism, this call can be used to make sure no other command is waiting ot run.

GetLastCommandError

Prototype: int GetLastCommandError(void)

Description: Returns an error value from the last pending command which had to wait for BadgeBuilder® to be in the main loop before executing.

Returned Data: The error code from the last pending command to execute. Use the GetErrorMessage call to get the text associated with the value.

GetFeatures

Prototype: XMLRECORDDATA * GetFeatures(void)

Description: Returns a structure with each element consisting of the feature name and its availability. See the appendix for more information on the list of features.

Returned Data:
NonTryoutFeature1 = TRUE
NonTryoutFeatureName1 = Dossier
NonTryoutFeature2 = TRUE
NonTryoutFeatureName2 = Magnetic Encoding
NonTryoutFeature3 = FALSE
NonTryoutFeatureName3 = No Video Capture
.
.
etc.

GetProgramInfo

Prototype: XMLRECORDDATA * GetProgramInfo(void)

Description: Returns a structure with each element consisting of the program information identifier and its value. The meaning of those values which are not self-explanatory are as follows:

PhotoCompressionType: 0=PCX, 1=RLE, 2=JPG, 3=FIF, 4=PNG
DossierIndex: -1 if undefined, else template index number
CurrentAccessLevel: 0=Browse, 1=Normal, 2=Supervisory, 3=Master, 4=Demo, 5=Server
RemainingTryoutCount: the number of times tryout features can be used before they become unavailable

Example Returned Data:
IDCFile = E:\BadgeBuilder\Testing\XML.idc
DatabaseType = ACCESS
PhotoCompressionType = 4
PhotoExtension = .PNG
PhotoPath = E:\BadgeBuilder\Testing\XML.~~~\
DossierIndex = -1
ProgramVersion = 4.50Beta
ProductVersion = 4, 50, 0, 0
CurrentAccessLevel = 0
CurrentAccessLevelName = BROWSE
RemainingTryoutCount = 16

GetRecord

Prototype: XMLRECORDDATA * GetRecord(RecordData *data)

Description: Returns a structure with each element consisting of the field name and its value for the record with the given datakey. The datakey is the value in the field defined as the index for the badge database.

Input Data Format:
Datakey = <recordkey>

Example Returned Data:
Datakey = 54312
Template = 0
Changed = X
FirstName = James
LastName = West
SignOn = 3/21/2002
BadgeNo = 5432

GetCurrent

Prototype: XMLRECORDDATA * GetCurrent(void)

Description: Returns a structure with each element consisting of the field name and its value for the currently selected record.

Example Returned Data:
Datakey = 54312
Template = 0
Changed = X
FirstName = James
LastName = West
SignOn = 3/21/2002
BadgeNo = 5432

GetCount

Prototype: XMLRECORDDATA * GetCount(void)

Description: Returns a structure with each element consisting of the values for the current record, number of selected records and total number of records. **CurrentIndex** is the index of the current badge in the selected badges. **Selected** is the number of badges currently selected and **Total** is the total number of badges.

Example Returned Data:
CurrentIndex = 3
Selected = 15
Total = 15

GetFields

Prototype: XMLRECORDDATA * GetFields(void)

Description: Returns a structure containing field names and related information associated with the database. The meaning of those field names which are not self-explanatory are as follows:

Type: 0=Alphanumeric, 1=Numeric, 2=Counter
AlphanumericType: -1=Not Alphanumeric Field, 0=General,1=Date, 2=Time, 3=Timestamp
Places: Decimal places for numeric type, maximum string length for alphanumeric type

Example Returned Data:
Datakey = Datakey
Photofile = PhotoFile
Template = Template
Dirty = Changed
UserField1Name = FirstName
UserField1Type = 0
UserField1AlphanumericType = 0
UserField1Places = 40
UserField1Alias = First Name
UserField2Name = LastName
UserField2Type = 0
UserField2AlphanumericType = 0
UserField2Places = 40
UserField2Alias = Last Name
UserField3Name = SignOn
UserField3Type = 0
UserField3AlphanumericType = 1
UserField3Places = 23
UserField3Alias = Sign On Date
UserField4Name = BadgeNo

GetTemplates

Prototype: XMLRECORDDATA * GetTemplates(void)

Description: Returns a structure containing the number of templates assigned to the open database, their indices and their names.

Example Returned Data:
Templates = 2
Template1Index = 0
Template1Name = xml
Template2Index = 1
Template2Name = test

GetStatus

Prototype: XMLRECORDDATA * GetStatus(void)

Description: Returns a structure containing the last error, the last pending command error, if a command is pending and if the program is in the idle loop.

Example Returned Data:
LastError = 0
LastPendingCommandError = 32
CommandPending = 0
InIdleLoop = 1

GetToolbarStatus

Prototype: XMLRECORDDATA * GetToolbarStatus(void)

Description: Returns a structure indicating the availability status of all the toolbar items. It determines the status of all the toolbar buttons at the actual time it is issued.

A 1 means enabled. This returns the status of all toolbar items, even though some may not be visit.

Example Returned Data:
ToolbarItemCount = 25
Open = 0
New = 0
Password = 1
New Record = 0
Edit Record = 0
Find First = 0
Find Previous = 0
Find Next = 1
Find Last = 1
Select All = 1
Fast Find = 1
Search = 1
Sort = 1
Verify = 1
Dossier = 0
Flip = 1
Video = 0
Acquire = 0
Bitmap = 0
Signature = 0
Fingeprint = 0
Delete = 0
Delete Selected = 0
Print = 0
Help = 1

GetBBHandle

Prototype: int GetBBHandle(void)

Description: Returns the handle of the BadgeBuilder® main window.

Returned Data: integer containing the Window handle of the BadgeBuilder® application.

GetEditCount

Prototype: int GetEditCount(void)

Description: Returns the number of edit windows in progress and not yet completed.

Returned Data: integer containing the number of currently open edit windows.

GetUserData

Prototype: XMLRECORDDATA * GetUserData(void)

Description: Returns a structure containing the User information.

Example Returned Data:
Company = XYZ Corp

Street = 999 East Street
State = WE
Post Code = 01234
Country = USA

GetBadgeImage

Prototype: HBITMAP GetBadgeImage(RecordData *data)

Description: This will create an HBITMAP handle to a BMP image containing the currently displayed badge. Either the front or back of the badge can be obtained.

ImageType determines how the file is transmitted via the XML link. Unless the XML data is being used directly or the http link is slow, the JPG form should not be used as poorer image quality may result.

BadgeBuilder® will be responsible for freeing the image and this will happen when another image of any type is requested. The client application should copy the bitmap if it intends to keep it.

Input Data Format:
ShowBack = <yes | 1 | true | on> any other value or missing equals false.
ImageType = <jpg | bmp> bmp is the default

Returned Data: Windows® handle to a bitmap

GetPhotoImage

Prototype: HBITMAP GetPhotoImage(RecordData *data)

Description: This will create an HBITMAP handle to a BMP image containing the indexed photo associated with the current badge, if there is one. The main photo has index 0 and sub-images are 1 ,2, etc.

ImageType determines how the file is transmitted via the XML link.

Unless the XML data is being used directly or the http link is slow the JPG form should not be used as poorer image quality may result.

BadgeBuilder® will be responsible for freeing the image and this will happen when another image of any type is requested. The client application should copy the bitmap if it intends to keep it.

Input Data Format:
ImageIndex = <photo index number>
ImageType = <jpg | bmp> bmp is the default

Returned Data: Windows® handle to a bitmap

GetSignatureImage

Prototype: HBITMAP GetSignatureImage(RecordData *data)

Description: This will create an HBITMAP handle to a BMP image containing the signature associated with the current badge, if there is one. ImageType determines how the file is transmitted via the XML link.

Unless the XML data is being used directly or the http link is slow the JPG form should not be used as poorer image quality may result.

BadgeBuilder® will be responsible for freeing the image and this will happen when another image of any type is requested. The client application should copy the bitmap if it intends to keep it.

Input Data Format:
ImageType = <jpg | bmp> bmp is the default

Returned Data: Windows® handle to a bitmap

GetFingerprintImage

Prototype: HBITMAP GetFingerprintImage(void)

Description: This will create an HBITMAP handle to a BMP image containing the fingerprint associated with the current badge, if there is one.

ImageType determines how the file is transmitted via the XML link. Unless the XML data is being used directly or the http link is slow the JPG form should not be used as poorer image quality may result.

BadgeBuilder® will be responsible for freeing the image and this will happen when another image of any type is requested. The client application should copy the bitmap if it intends to keep it.

Input Data Format:
ImageType = <jpg | bmp> bmp is the default

Returned Data: Windows® handle to a bitmap

Pending Commands

Many commands require BadgeBuilder® to take some specific action which requires it be at idle. For example, it cannot move to the next badge if it currently displaying the printing dialog. So, when a command is sent which requires BadgeBuilder® to be at idle it is stored and executed as soon as this condition is fullfilled, which may not be immediately. Thus the issued command becomes pending. Only one command can be pending at a time. There are various functions to help determine if a command is currently pending. Another command allows access to the error code returned by the pending command when it did execute.

All calls in the Informational Calls sections do not use the pending mechanism, and all those in the Operational Calls do.

Operational Calls

Some calls return a value of TriStateValue. This type is defined in **BBXMLClientDLLInterface.h** and has the following 3 possible values:

-1 = Error
0 = False
1 = True

When a call returns **bool** it returns **true** on success.

Open

Prototype: TriStateValue Open(struct data)

Description: This command will attempt to open the ".IDC" file given in the command. If a database is currently open, it will be closed, even if the open fails for the new database. The requested database must exist and the <IDCName> must be a fully qualified BadgeBuilder® database filename with the ".IDC" extension.

If the BadgeBuilder® application is minimized when this command is issued, the Information Pop Up window requesting the user to wait while the operation proceeds will NOT display.

Input Data Format:
IDCFileName = <existing IDC filename>

Close

Prototype: bool Close(void)

Description: This command close any database which is currently open, otherwise it will do nothing.

First

Prototype: bool First(void)

Description: This command selects the first record from the current selection in the database.

Next

Prototype: bool Next(void)

Description: This command selects the next record from the current selection in the database.

Previous

Prototype: bool Previous(void)

Description: This command selects the previous record from the current selection in the database.

Last

Prototype: bool Last(void)

Description: This command selects the last record from the current selection in the database.

Select

Prototype: bool Select(struct data)

Description: This command selects the records which match the supplied where clause. Just the text to the right of the "where" should be provided. Normal SQL rules apply to quoting text. Possible values of WhereClause are:

WhereClause= firstName='Jane'
WhereClause=firstName like 'Wil%'
WhereClause-firstName like 'B%' and BadgeNo > 1000

The second example returns all records where the firstName field starts with **Wil**. The third returns all with a firstName which begins with **B** and with a BadgeNo greater than **1000**.

Input Data Format:
WhereClause = <SQL where clause>

Delete

Prototype: bool Delete(struct data)

Description: This command deletes the record with the matching datakey. The parameter name **Datakey** may not be the actual field name of the datakey field.

Input Data Format:
Datakey = <recordkey>

Revise

Prototype: bool Revise(struct record)

Description: This command revises the contents of the record matching the datakey in the provided record. If no matching record exists it will **not** be created.

The parameter name **Datakey** may not be the actual field name of the datakey field and must be present. The remaining fields will cause the record's data to be changed if present. If a field is missing, existing data will remain unchanged.

Date, Time and Timestamp fields must have dates and times entered according to the format defined in the Windows® Control Panel.

Example Input Data Format:
Datakey = 54332
FirstName = Jane
LastName = West
SignOn = 3/21/2004
BadgeNo = 9988

Add

Prototype: bool Add(struct record)

Description: Adds the given record information to the database.

The parameter name **Datakey** may not be the actual field name of the datakey field and must be present. The provided datakey value must not already exist.

The remaining fields will cause the record's data to be changed if present. If a field is missing, the field will remain empty.

Date, Time and Timestamp fields must have dates and times entered according to the format defined in the Windows® Control Panel.

Example Input Data Format:
Datakey = 54332
FirstName = Jane
LastName = West
SignOn = 3/21/2004
BadgeNo = 9988

Update

Prototype: int Update(struct record)

Description: This command updates the contents of the record matching the datakey in the provided record. If no matching record exists it will be created.

The parameter name **Datakey** may not be the actual field name of the datakey field and must be present. The provided datakey value must not already exist.

The remaining fields will cause the record's data to be changed if present. If a field is missing, existing data will remain unchanged if the record already exists, and will be empty if a new record is created.

Date, Time and Timestamp fields must have dates and times entered according to the format defined in the Windows® Control Panel.

Example Input Data Format:
Datakey = 54332
FirstName = Jane
LastName = West
SignOn = 3/21/2004
BadgeNo = 9988

Exit

Prototype: bool Exit(void)

Description: This command Close the current database, if any, and terminates BadgeBuilder®.

Startlog

Prototype: bool Startlog(void)

Description: Turns BadgeBuilder® event logger on. The event logger feature in BadgeBuilder® writes all changes made to the database to an event file. Events captured are appended to the end of any existing log file.

Stoplog

Prototype: bool Stoplog(void)

Description: Turns BadgeBuilder® event logger off.

Restore

Prototype: bool Restore(void)

Description: Restores the BadgeBuilder® window and gives it the focus. This will restore it from either the maximized or minimized state.

Focus

Prototype: bool Focus(struct data)

Description: This command causes BadgeBuilder® to switch the window focus whenever any other pending command is completed and the main idle loop is again entered. To stop this effect, set the handle to zero.

Input Data Format:
Handle = <window handle to receive focus>

Video

Prototype: bool Video(void)

Description: This command causes BadgeBuilder® to open its video capture window the next time it returns to the main idle loop. It will only do so if the "Video" toolbar button is available and calling this command is equivalent to clicking on this tool button.

The video capture dialog will display even if the main BadgeBuilder® application is minimized. The BadgeBuilder® application will remain minimized during this command if it was originally minimized.

Print

Prototype: bool Print(struct data)

Description: If the “quick” parameter is YES printing will start immediately if conditions permit, using the current printing settings.

Otherwise, this command causes BadgeBuilder® to open its printing dialog window the next time it returns to the main idle loop. It will only do so if the "Printer" toolbar button is available and calling this command is equivalent to clicking on this tool button.

The printing dialog will display even if the main BadgeBuilder® application is minimized. The BadgeBuilder® application will remain minimized during this command if it was originally minimized.

Input Data Format:
Quick = <yes or no> defaults to No if missing

Twain

Prototype: bool Twain(void)

Description: This command causes BadgeBuilder® to open its TWAIN capture window the next time it returns to the main idle loop. It will only do so if the "TWAIN" toolbar button is available and calling this command is equivalent to clicking on this tool button.

The TWAIN capture dialog will display even if the main BadgeBuilder® application is minimized. The BadgeBuilder® application will remain minimized during this command if it was originally minimized.

Signature

Prototype: bool Signature(void)

Description: This command causes BadgeBuilder® to open its Signature capture window the next time it returns to the main idle loop. It will only do so if the "Signature" toolbar button is available and calling this command is equivalent to clicking on this tool button.

The Signature capture dialog will display even if the main BadgeBuilder® application is minimized. The BadgeBuilder® application will remain minimized during this command if it was originally minimized.

Bitmap

Prototype: bool Bitmap(void)

Description: This command causes BadgeBuilder® to open its Bitmap capture window the next time it returns to the main idle loop. It will only do so if the "Bitmap" toolbar button is available and calling this command is equivalent to clicking on this tool button.

The Bitmap capture dialog will display even if the main BadgeBuilder® application is minimized. The BadgeBuilder® application will remain minimized during this command if it was originally minimized.

Preview

Prototype: bool Preview(void)

Description: This command displays a preview of the badge. If more than one badge is currently selected then the navigation buttons will be enabled to permit browsing through the selected badges.

If double-sided badges are enabled, then the "flip" button will be enabled to permit previewing of the reverse side of the badge.

If BadgeBuilder® is in a mode, which permits printing, then the printer button will also be available. To the right of the print button is an exit button to allow the preview window to be closed without printing.

ServerMode

Prototype: bool ServerMode(struct toolbarSetting)

Description: Turns server mode on and off and optionally controls the toolbar elements shown when it is on.

When ServerMode is on, no actions can be taken by the user at the BadgeBuilder® window which would change the database in any way. In addition, if the mini-toolbar is active, the user will not be allowed to close the program.

When ServerMode is turned on, a mini-toolbar can be activated and optionally the contents of the toolbar defined. When this is done, the BadgeBuilder® windows reduces to just be large enough to contain the current badge. If no toobar items are defined for the mini-toolbar, then the following toolbar items will display:

Flip
Video
TWAIN
Bitmap
Signature
Fingerprint
Print
Help

although some may not be active if it not appropriate from them to be so.

Although all toolbar buttons can be displayed, many may not be appropriate in Server Mode. For example, the “User Log In” button will always be disabled in Server Mode, therefore displaying it serves no useful purpose.

When ServerMode is turned on, the mini-toolbar cannot be changed without and intervening command to turn ServerMode off.

When ServerMode and Minitoolbar are turned on, the BadgeBuilder® menu system is hidden.

Example Input Data Format:
servermode=on
minitoolbar=on
Find First = 1
Find Last = 1
Select All = 1
Fast Find = 1
Search = 1
Sort = 1
Verify = 1

Goto

Prototype: bool Goto(struct data)

Description: Select the nth record from the current selection in the database. Note that the index of the first badge is 1.

Input Data Format:
RecordNumber = <recordnumber> defaults to 1 if missing

Edit

Prototype: bool Edit(struct data)

Description: This command causes the BadgeBuilder® window to focus and execute a "Revise Current Badge Design" command.

When this command is received BadgeBuilder® remembers its current window state (minimized, normal, etc) and will return to that state when the edit operation is completed, as long as SERVERMODE is ON.

Input Data Format:
TemplateNumber = <valid template number>

CreateTemplate

Prototype: bool CreateTemplate(struct data)

Description: This command causes the BadgeBuilder® window to focus and execute a "Create New Template" command. A database need not open.

When this command is received BadgeBuilder® remembers its current window state (minimized, normal, etc) and will return to that state when the user operation is completed, as long as SERVERMODE is ON.

The FocusHandle is optional and is the handle of a window to receive the focus when the command completes.

Input Data Format:
TemplateStyle = <template style number> 0 - 6 are valid
FocusHandle = <window handle to receive focus after execution>

ReviseTemplate

Prototype: bool ReviseTemplate(struct data)

Description: This command causes the BadgeBuilder® window to focus and execute a Edit Existing Template command. A database need not open.

When this command is received BadgeBuilder® remembers its current window state (minimized, normal, etc) and will return to that state when the user operation is completed, as long as SERVERMODE is not OFF.

The FocusHandle is optional and is the handle of a window to receive the focus when the command completes.

Input Data Format:
FocusHandle = <window handle to receive focus after execution>

ManageTemplates

Prototype: bool ManageTemplates(void)

Description: This command causes the BadgeBuilder® window to focus and execute a "Manage Database Templates" command. A database must be open.

When this command is received BadgeBuilder® remembers its current window state (minimized, normal, etc) and will return to that state when the user operation is completed, as long as SERVERMODE is not OFF.

ReviseAssignments

Prototype: bool ReviseAssignments(struct data)

Description: This command causes the BadgeBuilder® window to focus and execute a "Reassign Field Assignments" command. A database must be open.

When this command is received BadgeBuilder® remembers its current window state (minimized, normal, etc) and will return to that state when the user operation is completed, as long as [SERVERMODE] is not OFF.

Input Data Format:
Index = <template number> -1 is current template, defaults to -1

Fingerprint

Prototype: bool Fingerprint(void)

Description: This command causes BadgeBuilder® to open its Fingerprint capture window the next time it returns to the main idle loop.

It will only do so if the "Fingerprint" toolbar button is available and calling this command is equivalent to clicking on this tool button.

The Fingerprint capture dialog will display even if the main BadgeBuilder® application is minimized. The BadgeBuilder® application will remain minimized during this command if it was originally minimized.

Template

Prototype: bool Template(struct data)

Description: This command causes the currently selected badge to have its template set to the given index. The new appearance of the badge will also display immediately.

Input Data Format:
Index = <template number>

ReviseUserData

Prototype: bool ReviseUserData(struct data)

Description: This will change the user data using the data provided. If a field is omitted the existing data will remain unchanged.

Input Data Example:
Company = XYZ Corp
Street = 999 East Street

FlipBadge

Prototype: bool FlipBadge(struct data)

Description: This will show the front or back of the badge depending upon the command in the passed data.

For this command to be successful, BadgeBuilder® must have the double-sided feature available to it.

Input Data Format:
ShowBack = <yes | 1 | true | on> any other value or missing equals false.

TeamTracerConfig

Prototype: bool TeamTracerConfig(void)

Description: This will show the TeamTracer configuration window if a database is open..

For this command to be successful, TeamTracer® and not BadgeBuilder® must be running and a database open. BadgeBuilder® will return a "Feature not available" error.

TeamTracer

Prototype: bool TeamTracer(void)

Description: When TeamTracer® is the target application, this will show the TeamTracer event window if a database is open and configured for TeamTracer operation. BadgeBuilder® will return a "Feature not available" error.

XML Interface

Introduction

An XML client can be written in any language which supports HTML communications with an XML server. This includes languages like PHP, a popular scripting language (see www.php.net).

The following sections displays by example the posted XML command and an example of the returned XML response.

See the section called DLL Interface for a description of the functionality of each command.

Note that every XML post must include a Key field containing the key currently set in BadgeBuilder®. This is the default key shown in these examples unless changed by the bb32.ini file.

XML Error Return

If an error occurs when a call is made, a standard XML response will occur. This is an example of a XML error return:

```
<?xml version="1.0"?>
<methodResponse>
  <fault>
    <value>
     <struct>
        <member>
          <name>faultCode</name>
          <value><int>13</int></value>
          </member>
        <member>
          <name>faultString</name>
          <value><string>Already at first record</string></value>
          </member>
        </struct>
      </value>
    </fault>
  </methodResponse>
```

Informational Calls

GetErrorMessage

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>GetErrorMessage</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>errorcode</name>
   <value><string>34</string></value>
  </member>
  <member>
   <name>BBXMLRPCKey</name>
  <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><string>Update failed</string></value>
  </param>
  </params>
</methodResponse>
```

IsBusy

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>IsBusy</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
```

```
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>0</boolean></value>
  </param>
  </params>
</methodResponse>
```

GetLastCommandError

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>GetLastCommandError</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><int>0</int></value>
  </param>
  </params>
</methodResponse>
```

GetFeatures

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>GetFeatures</methodName>
```

```
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>NonTryoutFeature1</name>
   <value><string>TRUE</string></value>
  </member>
  <member>
   <name>NonTryoutFeatureName1</name>
   <value><string>Dossier</string></value>
  </member>
  <member>
   <name>NonTryoutFeature2</name>
   <value><string>TRUE</string></value>
  </member>
  <member>
   <name>NonTryoutFeatureName2</name>
   <value><string>Magnetic Encoding</string></value>
  </member>
  <member>
   <name>NonTryoutFeature3</name>
   <value><string>FALSE</string></value>
  </member>
  <member>
   <name>NonTryoutFeatureName3</name>
   <value><string>No Video Capture</string></value>
  </member>
  <member>
   <name>NonTryoutFeature4</name>
   <value><string>FALSE</string></value>
  </member>
  <member>
   <name>NonTryoutFeatureName4</name>
   <value><string>No Database Creation</string></value>
  </member>
```

```
<member>
 <name>NonTryoutFeature5</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName5</name>
 <value><string>Skin Tone Control</string></value>
</member>
<member>
 <name>NonTryoutFeature6</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName6</name>
 <value><string>Browse Only</string></value>
</member>
<member>
 <name>NonTryoutFeature7</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName7</name>
 <value><string>Single Photo</string></value>
</member>
<member>
 <name>NonTryoutFeature8</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName8</name>
 <value><string>Fingerprint Enrollment</string></value>
</member>
<member>
 <name>NonTryoutFeature9</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName9</name>
 <value><string>BBDirect Double Sided</string></value>
</member>
<member>
 <name>NonTryoutFeature10</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName10</name>
 <value><string>BBDirect Video</string></value>
</member>
<member>
 <name>NonTryoutFeature11</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName11</name>
 <value><string>BBDirect Options</string></value>
</member>
```

```
<member>
 <name>NonTryoutFeature12</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName12</name>
 <value><string>Visitor Only</string></value>
</member>
<member>
 <name>NonTryoutFeature13</name>
 <value><string>TRUE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName13</name>
 <value><string>Smart Cards</string></value>
</member>
<member>
 <name>NonTryoutFeature14</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName14</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>NonTryoutFeature15</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName15</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>NonTryoutFeature16</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName16</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>NonTryoutFeature17</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName17</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>NonTryoutFeature18</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName18</name>
 <value><string>Undefined</string></value>
</member>
```

```
<member>
 <name>NonTryoutFeature19</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName19</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>NonTryoutFeature20</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName20</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>NonTryoutFeature21</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName21</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>NonTryoutFeature22</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName22</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>NonTryoutFeature23</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName23</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>NonTryoutFeature24</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName24</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>NonTryoutFeature25</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName25</name>
 <value><string>Undefined</string></value>
</member>
```

```
<member>
 <name>NonTryoutFeature26</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName26</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>NonTryoutFeature27</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName27</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>NonTryoutFeature28</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName28</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>NonTryoutFeature29</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName29</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>NonTryoutFeature30</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName30</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>NonTryoutFeature31</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName31</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>NonTryoutFeature32</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>NonTryoutFeatureName32</name>
 <value><string>VisitorPrimary</string></value>
</member>
```

```
<member>
  <name>TryoutFeature1</name>
  <value><string>TRUE</string></value>
</member>
<member>
  <name>TryoutFeatureName1</name>
  <value><string>TWAIN Interface</string></value>
</member>
<member>
  <name>TryoutFeature2</name>
  <value><string>TRUE</string></value>
</member>
<member>
  <name>TryoutFeatureName2</name>
  <value><string>Signature Capture</string></value>
</member>
<member>
  <name>TryoutFeature3</name>
  <value><string>TRUE</string></value>
</member>
<member>
  <name>TryoutFeatureName3</name>
  <value><string>Reports</string></value>
</member>
<member>
  <name>TryoutFeature4</name>
  <value><string>TRUE</string></value>
</member>
<member>
  <name>TryoutFeatureName4</name>
  <value><string>Photo Capture from Bitmap</string></value>
</member>
<member>
  <name>TryoutFeature5</name>
  <value><string>TRUE</string></value>
</member>
<member>
  <name>TryoutFeatureName5</name>
  <value><string>Double-sided Badges</string></value>
</member>
<member>
  <name>TryoutFeature6</name>
  <value><string>TRUE</string></value>
</member>
<member>
  <name>TryoutFeatureName6</name>
  <value><string>Verifier</string></value>
</member>
<member>
  <name>TryoutFeature7</name>
  <value><string>TRUE</string></value>
</member>
<member>
  <name>TryoutFeatureName7</name>
  <value><string>Video for Windows Capture</string></value>
</member>
```

```
<member>
 <name>TryoutFeature8</name>
 <value><string>TRUE</string></value>
</member>
<member>
 <name>TryoutFeatureName8</name>
 <value><string>TeamTracer</string></value>
</member>
<member>
 <name>TryoutFeature9</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName9</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>TryoutFeature10</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName10</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>TryoutFeature11</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName11</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>TryoutFeature12</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName12</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>TryoutFeature13</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName13</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>TryoutFeature14</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName14</name>
 <value><string>Undefined</string></value>
</member>
```

```
<member>
 <name>TryoutFeature15</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName15</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>TryoutFeature16</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName16</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>TryoutFeature17</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName17</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>TryoutFeature18</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName18</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>TryoutFeature19</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName19</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>TryoutFeature20</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName20</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>TryoutFeature21</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName21</name>
 <value><string>Undefined</string></value>
</member>
```

```
<member>
 <name>TryoutFeature22</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName22</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>TryoutFeature23</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName23</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>TryoutFeature24</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName24</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>TryoutFeature25</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName25</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>TryoutFeature26</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName26</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>TryoutFeature27</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName27</name>
 <value><string>Undefined</string></value>
</member>
<member>
 <name>TryoutFeature28</name>
 <value><string>FALSE</string></value>
</member>
<member>
 <name>TryoutFeatureName28</name>
 <value><string>Undefined</string></value>
</member>
```

```
    <member>
      <name>TryoutFeature29</name>
      <value><string>FALSE</string></value>
    </member>
    <member>
      <name>TryoutFeatureName29</name>
      <value><string>Undefined</string></value>
    </member>
    <member>
      <name>TryoutFeature30</name>
      <value><string>FALSE</string></value>
    </member>
    <member>
      <name>TryoutFeatureName30</name>
      <value><string>Undefined</string></value>
    </member>
    <member>
      <name>TryoutFeature31</name>
      <value><string>FALSE</string></value>
    </member>
    <member>
      <name>TryoutFeatureName31</name>
      <value><string>Undefined</string></value>
    </member>
    <member>
      <name>TryoutFeature32</name>
      <value><string>FALSE</string></value>
    </member>
    <member>
      <name>TryoutFeatureName32</name>
      <value><string>Undefined</string></value>
    </member>
  </struct>
</value>

   </param>
   </params>
</methodResponse>
```

GetProgramInfo

XML Post

```
<?xml version="1.0"?>
<methodCall>
   <methodName>GetProgramInfo</methodName>
   <params>
   <param>
<value>
  <struct>
    <member>
      <name>BBXMLRPCKey</name>
      <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
    </member>
  </struct>
```

```
</value>

  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>IDCFile</name>
<value><string>E:\BadgeBuilder\Testing\XML.idc</string></value>
  </member>
  <member>
   <name>DatabaseType</name>
   <value><string>ACCESS</string></value>
  </member>
  <member>
   <name>PhotoCompressionType</name>
   <value><string>4</string></value>
  </member>
  <member>
   <name>PhotoExtension</name>
   <value><string>.PNG</string></value>
  </member>
  <member>
   <name>PhotoPath</name>
<value><string>E:\BadgeBuilder\Testing\XML.~~~\</string></value>
  </member>
  <member>
   <name>DossierIndex</name>
   <value><string>-1</string></value>
  </member>
  <member>
   <name>ProgramVersion</name>
   <value><string>4.50Beta</string></value>
  </member>
  <member>
   <name>ProductVersion</name>
   <value><string>4, 50, 0, 0</string></value>
  </member>
  <member>
   <name>CurrentAccessLevel</name>
   <value><string>5</string></value>
  </member>
  <member>
   <name>CurrentAccessLevelName</name>
   <value><string>Server</string></value>
  </member>
  <member>
   <name>RemainingTryoutCount</name>
```

```
    <value><string>20</string></value>
   </member>
  </struct>
</value>
  </param>
  </params>
</methodResponse>
```

GetRecord

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>GetRecord</methodName>
  <params>
  <param>
<value>
  <struct>
   <member>
    <name>datakey</name>
    <value><string>54332</string></value>
   </member>
   <member>
    <name>BBXMLRPCKey</name>
    <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
   </member>
  </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value>
  <struct>
   <member>
    <name>Datakey</name>
    <value><string>54332</string></value>
   </member>
   <member>
    <name>PhotoFile</name>
    <value><string>RCLHSYSB.PNG</string></value>
   </member>
   <member>
    <name>Template</name>
    <value><string>0</string></value>
   </member>
   <member>
    <name>Changed</name>
    <value><string>X</string></value>
```

```
    </member>
    <member>
      <name>FirstName</name>
      <value><string>Jane</string></value>
    </member>
    <member>
      <name>LastName</name>
      <value><string>West</string></value>
    </member>
    <member>
      <name>SignOn</name>
      <value><string>3/21/2004</string></value>
    </member>
    <member>
      <name>BadgeNo</name>
      <value><string>9988</string></value>
    </member>
  </struct>
</value>
   </param>
   </params>
</methodResponse>
```

GetCurrent

XML Post

```
<?xml version="1.0"?>
<methodCall>
   <methodName>GetCurrent</methodName>
   <params>
   <param>
<value>
  <struct>
    <member>
      <name>BBXMLRPCKey</name>
      <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
    </member>
  </struct>
</value>
   </param>
   </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
   <params>
   <param>
<value>
  <struct>
    <member>
      <name>Datakey</name>
      <value><string>54332</string></value>
    </member>
```

```
    <member>
      <name>PhotoFile</name>
      <value><string>RCLHSYSB.PNG</string></value>
    </member>
    <member>
      <name>Template</name>
      <value><string>0</string></value>
    </member>
    <member>
      <name>Changed</name>
      <value><string>X</string></value>
    </member>
    <member>
      <name>FirstName</name>
      <value><string>Jane</string></value>
    </member>
    <member>
      <name>LastName</name>
      <value><string>West</string></value>
    </member>
    <member>
      <name>SignOn</name>
      <value><string>3/21/2004</string></value>
    </member>
    <member>
      <name>BadgeNo</name>
      <value><string>9988</string></value>
    </member>
  </struct>
</value>
   </param>
   </params>
</methodResponse>
```

GetCount

XML Post

```
<?xml version="1.0"?>
<methodCall>
   <methodName>GetCount</methodName>
   <params>
   <param>
<value>
  <struct>
    <member>
      <name>BBXMLRPCKey</name>
      <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
    </member>
  </struct>
</value>
   </param>
   </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>CurrentIndex</name>
   <value><string>1</string></value>
  </member>
  <member>
   <name>Selected</name>
   <value><string>15</string></value>
  </member>
  <member>
   <name>Total</name>
   <value><string>15</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodResponse>
```

GetFields

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>GetFields</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value>
 <struct>
  <member>
```

```
 <name>Datakey</name>
 <value><string>Datakey</string></value>
</member>
<member>
 <name>Photofile</name>
 <value><string>PhotoFile</string></value>
</member>
<member>
 <name>Template</name>
 <value><string>Template</string></value>
</member>
<member>
 <name>Dirty</name>
 <value><string>Changed</string></value>
</member>
<member>
 <name>UserField1Name</name>
 <value><string>FirstName</string></value>
</member>
<member>
 <name>UserField1Type</name>
 <value><string>0</string></value>
</member>
<member>
 <name>UserField1AlphanumericType</name>
 <value><string>0</string></value>
</member>
<member>
 <name>UserField1Places</name>
 <value><string>40</string></value>
</member>
<member>
 <name>UserField1Alias</name>
 <value><string>First Name</string></value>
</member>
<member>
 <name>UserField2Name</name>
 <value><string>LastName</string></value>
</member>
<member>
 <name>UserField2Type</name>
 <value><string>0</string></value>
</member>
<member>
 <name>UserField2AlphanumericType</name>
 <value><string>0</string></value>
</member>
<member>
 <name>UserField2Places</name>
 <value><string>40</string></value>
</member>
<member>
 <name>UserField2Alias</name>
 <value><string>Last Name</string></value>
</member>
<member>
```

```
    <name>UserField3Name</name>
    <value><string>SignOn</string></value>
   </member>
   <member>
    <name>UserField3Type</name>
    <value><string>0</string></value>
   </member>
   <member>
    <name>UserField3AlphanumericType</name>
    <value><string>1</string></value>
   </member>
   <member>
    <name>UserField3Places</name>
    <value><string>23</string></value>
   </member>
   <member>
    <name>UserField3Alias</name>
    <value><string>Sign On Date</string></value>
   </member>
   <member>
    <name>UserField4Name</name>
    <value><string>BadgeNo</string></value>
   </member>
   <member>
    <name>UserField4Type</name>
    <value><string>1</string></value>
   </member>
   <member>
    <name>UserField4AlphanumericType</name>
    <value><string>-1</string></value>
   </member>
   <member>
    <name>UserField4Places</name>
    <value><string>0</string></value>
   </member>
   <member>
    <name>UserField4Alias</name>
    <value><string>Badge Number</string></value>
   </member>
  </struct>
</value>
   </param>
   </params>
</methodResponse>
```

GetTemplates

XML Post

```
<?xml version="1.0"?>
<methodCall>
   <methodName>GetTemplates</methodName>
   <params>
   <param>
<value>
```

```
  <struct>
   <member>
    <name>BBXMLRPCKey</name>
    <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
   </member>
  </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value>
  <struct>
   <member>
    <name>Templates</name>
    <value><string>2</string></value>
   </member>
   <member>
    <name>Template1Index</name>
    <value><string>0</string></value>
   </member>
   <member>
    <name>Template1Name</name>
    <value><string>xml</string></value>
   </member>
   <member>
    <name>Template2Index</name>
    <value><string>1</string></value>
   </member>
   <member>
    <name>Template2Name</name>
    <value><string>test</string></value>
   </member>
  </struct>
</value>
  </param>
  </params>
</methodResponse>
```

GetStatus

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>GetStatus</methodName>
  <params>
  <param>
<value>
  <struct>
```

```
   <member>
    <name>BBXMLRPCKey</name>
    <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
   </member>
  </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value>
  <struct>
   <member>
    <name>LastError</name>
    <value><string>0</string></value>
   </member>
   <member>
    <name>LastPendingCommandError</name>
    <value><string>0</string></value>
   </member>
   <member>
    <name>CommandPending</name>
    <value><string>0</string></value>
   </member>
   <member>
    <name>InIdleLoop</name>
    <value><string>1</string></value>
   </member>
  </struct>
</value>
  </param>
  </params>
</methodResponse>
```

GetToolbarStatus

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>GetToolbarStatus</methodName>
  <params>
  <param>
<value>
  <struct>
   <member>
    <name>BBXMLRPCKey</name>
    <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
   </member>
  </struct>
```

```
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>ToolbarItemCount</name>
   <value><string>25</string></value>
  </member>
  <member>
   <name>Open</name>
   <value><string>0</string></value>
  </member>
  <member>
   <name>New</name>
   <value><string>0</string></value>
  </member>
  <member>
   <name>Password</name>
   <value><string>0</string></value>
  </member>
  <member>
   <name>New Record</name>
   <value><string>0</string></value>
  </member>
  <member>
   <name>Edit Record</name>
   <value><string>0</string></value>
  </member>
  <member>
   <name>Find First</name>
   <value><string>0</string></value>
  </member>
  <member>
   <name>Find Previous</name>
   <value><string>0</string></value>
  </member>
  <member>
   <name>Find Next</name>
   <value><string>1</string></value>
  </member>
  <member>
   <name>Find Last</name>
   <value><string>1</string></value>
  </member>
  <member>
   <name>Select All</name>
   <value><string>1</string></value>
```

```
</member>
<member>
  <name>Fast Find</name>
  <value><string>1</string></value>
</member>
<member>
  <name>Search</name>
  <value><string>1</string></value>
</member>
<member>
  <name>Sort</name>
  <value><string>1</string></value>
</member>
<member>
  <name>Verify</name>
  <value><string>1</string></value>
</member>
<member>
  <name>Dossier</name>
  <value><string>0</string></value>
</member>
<member>
  <name>Flip</name>
  <value><string>1</string></value>
</member>
<member>
  <name>Video</name>
  <value><string>1</string></value>
</member>
<member>
  <name>Acquire</name>
  <value><string>1</string></value>
</member>
<member>
  <name>Bitmap</name>
  <value><string>0</string></value>
</member>
<member>
  <name>Signature</name>
  <value><string>1</string></value>
</member>
<member>
  <name>Fingeprint</name>
  <value><string>1</string></value>
</member>
<member>
  <name>Delete</name>
  <value><string>0</string></value>
</member>
<member>
  <name>Delete Selected</name>
  <value><string>0</string></value>
</member>
<member>
  <name>Print</name>
  <value><string>1</string></value>
```

```
    </member>
    <member>
      <name>Help</name>
      <value><string>1</string></value>
    </member>
  </struct>
</value>
  </param>
  </params>
</methodResponse>
```

GetBBHandle

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>GetBBHandle</methodName>
  <params>
  <param>
<value>
  <struct>
    <member>
      <name>BBXMLRPCKey</name>
      <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
    </member>
  </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><int>459206</int></value>
  </param>
  </params>
</methodResponse>
```

GetEditCount

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>GetEditCount</methodName>
  <params>
  <param>
<value>
  <struct>
    <member>
```

```
    <name>BBXMLRPCKey</name>
    <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
   </member>
  </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><int>0</int></value>
  </param>
  </params>
</methodResponse>
```

GetUserData

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>GetUserData</methodName>
  <params>
  <param>
<value>
  <struct>
   <member>
    <name>BBXMLRPCKey</name>
    <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
   </member>
  </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value>
  <struct>
   <member>
    <name>Company</name>
    <value><string>XYZ Corp</string></value>
   </member>
   <member>
    <name>Street</name>
    <value><string>999 East Street</string></value>
```

```
    </member>
    <member>
      <name>City</name>
      <value><string></string></value>
    </member>
    <member>
      <name>State</name>
      <value><string>WE</string></value>
    </member>
    <member>
      <name>Post Code</name>
      <value><string>01234</string></value>
    </member>
    <member>
      <name>Country</name>
      <value><string>USA</string></value>
    </member>
  </struct>
</value>
   </param>
   </params>
</methodResponse>
```

GetBadgeImage

XML Post

```
<?xml version="1.0"?>
<methodCall>
   <methodName>GetBadgeImage</methodName>
   <params>
   <param>
<value>
  <struct>
    <member>
      <name>datakey</name>
      <value><string>54332</string></value>
    </member>
    <member>
      <name>BBXMLRPCKey</name>
      <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
    </member>
  </struct>
</value>
   </param>
   </params>
</methodCall>
```

XML Return

The middle section of the returned encoded image has been omitted. Note also that each block of data ends with a line feed so any decoding software must ignore these.

```
<?xml version="1.0"?>
<methodResponse>
   <params>
```

```
  <param>
<value><base64>Qk32CAQAAAAAADYAAAAoAAAARAEAAMwAAAABAC
AAAAAAAAMAIBAAAAAAAAAAAAAAAAAAAAAAAA////
AP///wD///8A////AP///wD///8A////AP///wD///8A////AP///wD///8A
/wAAAP8AAAD/AAAA/wAAAP8AAAD/AAAA/wAAAP8AAAD/AAAA/wAAA
P8AAAD/AAAA/wAAAP8AAAD/
AAAA/wAAAP8AAAD/AAAA/wAAAP8AAAD/AAAA/wAAAP8AAAD/AAAA/
wAAAP8AAAD/AAAA/wAAAP8A
AAD/AAAA/wAAAP8AAAD/AAAA/wAAAP8AAAD/AAAA/wAAAP8AAAD/A
AAA/wAAAP8AAAD/AAAA/wAA
AP8AAAD/AAAA/wAAAP8AAAD/AAAA/wAAAP8AAAD/AAAA/wAAAP8AA
AD/AAAA/wAAAP8AAAD/AAAA
.
.
.
AAD/AAAA/wAAAP8AAAD/AAAA/wAAAP8AAAD/AAAA/wAAAP8AAAD/A
AAA/wAAAP8AAAD/AAAA/wAA
AP8AAAD/AAAA/wAAAP8AAAD/AAAA/wAAAP8AAAD/AAAA/wAAAP8AA
AD/AAAA/wAAAP8AAAD/AAAA
/wAAAP8AAAD/AAAA////AP///wD///8A////AP///wD///8A////AP///wD///8A////AP///
wD/
//8A////AP///wD///8A</base64></value>
  </param>
  </params>
</methodResponse>
```


GetPhotoImage

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>GetPhotoImage</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>datakey</name>
   <value><string>54332</string></value>
  </member>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

The middle section of the returned encoded image has been omitted. Note also that each block of data ends with a line feed so any decoding software must ignore these.

<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><base64>Qk1+tyMAAAAAADYAAAAoAAAAzgIAAC8DAAABACAA
AAAAAEi3IwAAAAAAAAAAAAAAAAAAAAAAAAX3R0
AF1ycgBlcXQAZ3N2AGZ0eQBmdHkAYnJ3AGFxdgBgcHcAYHB3AF9vdgBicnk
AYXN7AGByegBecnoA
YHR8AGF1fQBhdX0AZHp+AGF3ewBedHgAXXN3AGF4eQBlfH0AYnt8AF53e
ABeeXwAX3p9AF14fQBb
dnsAXXR7AF10ewBidHwAY3V9AGBzeQBidXsAZnd8AGZ3fABkdngAY3V3A
GV2eABneHoAZXZ3AGZ3
eABld3kAZHZ4AF92dwBfdncAYXd7AGN5fQBieHwAZHp+AGF5fQBje38AZoG
GAGWAhQBge4AAYXyB
AF96fwBfen8AZHyBAGN7gABken4AZnyAAGt9fwBneXsAY3Z8AGJ1ewBhdXk
AY3d7AGV3eQBkdngA
Y3R2AGJzdQBld3kAZXd5AGN3ewBjd3sAYXZ8AGJ3fQBheX4AYXl+AGR8gA
BlfYEAZH+CAGN+gQBd
e38AXHp+AFt7fwBefoIAXn6CAFt7fwBaen4AW3t/AF17fwBffYEAY36DAGB7g
ABhe4IAYnyDAGZ9
hABmfYQAaXuDAGh6ggBoe4EAaHuBAGZ5fwBmeX8AZHp+AGN5fQBmeHoA
ZXd5AGZ3eQBldngAZHV6
.
.
.
kbniAI623wCJtN0AjLfgAIm33wCHtd0AiLPXAIy32wCHsNUAiLHWAIWt1ACK
stkAjbXcAI213ACU
vOUAjbXeAIqz3ACJstsAi7fdAIq23ACPveMAiLbcAIuw0wCJrtEAiqzSAIqs0gCN
qtUAka7ZAJWy
3QCVst0AjarVAIyp1ACMqM8Ai6fOAISdwAB+l7oAdo6wAHCIqgB9lq8AhJ22AI
aiuwCLp8AAhKnC
AHmetwBola8AaZawAG+evABtnLoAbJ28AG6fvgA=</base64></value>
  </param>
  </params>
</methodResponse>

GetSignatureImage

XML Post

<?xml version="1.0"?>
<methodCall>
  <methodName>GetSignatureImage</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>datakey</name>
   <value><string>54332</string></value>
  </member>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>

```
  </struct>
</value>
   </param>
   </params>
</methodCall>
```

XML Return

The middle section of the returned encoded image has been omitted. Note also that each block of data ends with a line feed so any decoding software must ignore these.

```
<?xml version="1.0"?>
<methodResponse>
   <params>
   <param>
<value><base64>Qk32egQAAAAAADYAAAAoAAAAEAIAAIsAAAABACAAA
AAAAMB6BAAAAAAAAAAAAAAAAAAAAAAA////
AP///wD///8A////AP///wD///8A////AP///wD///8A////AP///wD///8A
////AP///wD///8A////AP///wD///8A////AP///wD///8A////AP///wD/
//8A////AP///wD///8A////AP///wD///8A////AP///wD///8A////AP//
/wD///8A////AP///wD///8A////AP///wD///8A////AP///wD///8A////
AP///wD///8A////AP///wD///8A////AP///wD///8A////AP///wD///8A
////AP///wD///8A////AP///wD///8A////AP///wD///8A////AP///wD/
//8A////AP///wD///8A////AP///wD///8A////AP///wD///8A////AP//
.
.
.
AP///wD///8A////AP///wD///8A////AP///wD///8A////AP///wD///8A
////AP///wD///8A////AP///wD///8A////AP///wD///8A////AP///wD/
//8A////AP///wD///8A</base64></value>
   </param>
   </params>
</methodResponse>
```

GetFingerprintImage

XML Post

```
<?xml version="1.0"?>
<methodCall>
   <methodName>GetFingerprintImage</methodName>
   <params>
   <param>
<value>
  <struct>
    <member>
      <name>datakey</name>
      <value><string>54332</string></value>
    </member>
    <member>
      <name>BBXMLRPCKey</name>
      <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
    </member>
  </struct>
</value>
   </param>
```

```
  </params>
</methodCall>
```

XML Return

The middle section of the returned encoded image has been omitted. Note also that each block of data ends with a line feed so any decoding software must ignore these.

```
<?xml version="1.0"?>

<methodResponse>

  <params>

  <param>

<value><base64>Qk02kAEAAAAAADYAAAAoAAAAoAAAAKAAAAABACA
AAAAAAACQAQAAAAAAAAAAAAAAAAAAAAAAAhoaG

AJaWlgCWlpYAhoaGAHd3dwB3d3cAmWZmAIaGhgCGhoYAmWZmAIaGhgCZ
ZmYAd3d3AHd3dwCWlpYA

hoaGAIaGhgB3d3cAd3d3AJlmmQCWlpYAhoaGAJaWlgCZZmYAhoaGAJaWlgC
WlpYAhoaGAJaWlgCG

hoYAlpaWAIaGhgCWlpYAlpaWAHd3dwCGhoYAlpaWAIaGhgCWlpYAlpaWAJ
aWlgCWlpYAhoaGAJaW

lgCGhoYAlpaWAIaGhgCGhoYAhoaGAJlmZgB3d3cAmWZmAHd3dwB3d3cAm
WZmAHd3dwCZZmYAZmZm
.
.
.
hoYAhoaGAIaGhgCWlpYAlpaWAJaWlgCWlpYAlpaWAIaGhgCGhoYAd3d3AJa
WlgCWlpYAlpaWAJaW
lgCWlpYAlpaWAJaWlgCWlpYAlpaWAJaWlgCGhoYAhoaGAIaGhgCWlpYAlpa
WAJaWlgCWlpYAlpaW
AJaWlgCWlpYAlpaWAIaGhgB3d3cAd3d3AA==</base64></value>
  </param>
  </params>
</methodResponse>
```

Operational Calls

Open

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Open</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>idcfilename</name>
```

```
    <value><string>e:\badgebuilder\testing\xml.idc</string></value>
   </member>
   <member>
    <name>BBXMLRPCKey</name>
    <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
   </member>
  </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Close

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Close</methodName>
  <params>
  <param>
<value>
  <struct>
   <member>
    <name>BBXMLRPCKey</name>
    <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
   </member>
  </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

First

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>First</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Next

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Next</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Previous

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Previous</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Last

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Last</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
```

```
    <name>BBXMLRPCKey</name>
    <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
   </member>
  </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Select

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Select</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>whereclause</name>
   <value><string>firstname like &amp;apos;J%&amp;apos;</string></value>
  </member>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Delete

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Delete</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>Datakey</name>
   <value><string>54352</string></value>
  </member>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Revise

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Revise</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>Datakey</name>
   <value><string>54332</string></value>
  </member>
  <member>
   <name>FirstName</name>
   <value><string>Janet</string></value>
  </member>
```

```
    <member>
      <name>LastName</name>
      <value><string>West</string></value>
    </member>
    <member>
      <name>SignOn</name>
      <value><string>3/21/2004</string></value>
    </member>
    <member>
      <name>BadgeNo</name>
      <value><string>9988</string></value>
    </member>
    <member>
      <name>BBXMLRPCKey</name>
      <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
    </member>
  </struct>
</value>
   </param>
   </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
   <params>
   <param>
<value><boolean>1</boolean></value>
   </param>
   </params>
</methodResponse>
```

Add

XML Post

```
<?xml version="1.0"?>
<methodCall>
   <methodName>Add</methodName>
   <params>
   <param>
<value>
  <struct>
    <member>
      <name>Datakey</name>
      <value><string>54632</string></value>
    </member>
    <member>
      <name>FirstName</name>
      <value><string>Janet</string></value>
    </member>
    <member>
      <name>LastName</name>
```

```
    <value><string>West</string></value>
   </member>
   <member>
    <name>SignOn</name>
    <value><string>3/21/2004</string></value>
   </member>
   <member>
    <name>BadgeNo</name>
    <value><string>9988</string></value>
   </member>
   <member>
    <name>BBXMLRPCKey</name>
    <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
   </member>
  </struct>
</value>

  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Update

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Update</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>Datakey</name>
   <value><string>54632</string></value>
  </member>
  <member>
   <name>FirstName</name>
   <value><string>Jane</string></value>
  </member>
  <member>
   <name>LastName</name>
   <value><string>West</string></value>
```

```
    </member>
    <member>
      <name>SignOn</name>
      <value><string>3/21/2004</string></value>
    </member>
    <member>
      <name>BadgeNo</name>
      <value><string>9988</string></value>
    </member>
    <member>
      <name>BBXMLRPCKey</name>
      <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
    </member>
  </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Exit

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Exit</methodName>
  <params>
  <param>
<value>
  <struct>
    <member>
      <name>BBXMLRPCKey</name>
      <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
    </member>
  </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
```

```
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Startlog

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Startlog</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
    <name>BBXMLRPCKey</name>
    <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Stoplog

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Stoplog</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
    <name>BBXMLRPCKey</name>
    <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
```

```
  </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Restore

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Restore</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Focus

XML Post

```
<?xml version="1.0"?>
<methodCall>
```

```
  <methodName>Focus</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>Handle</name>
   <value><string>564346</string></value>
  </member>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Video

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Video</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
```

```
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Print

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Print</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>Quick</name>
   <value><string>yes</string></value>
  </member>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Twain

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Twain</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>BBXMLRPCKey</name>
```

```
    <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
   </member>
  </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Signature

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Signature</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Bitmap

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Bitmap</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Preview

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Preview</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

ServerMode

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>ServerMode</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>ServerMode</name>
   <value><string>on</string></value>
  </member>
  <member>
   <name>Minitoolbar</name>
   <value><string>1</string></value>
  </member>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Goto

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Goto</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>RecordNumber</name>
   <value><string>4</string></value>
  </member>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Edit

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Edit</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>TemplateNumber</name>
   <value><string>0</string></value>
  </member>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
```

```
  </struct>
</value>

   </param>
   </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
   <params>
   <param>
<value><boolean>1</boolean></value>
   </param>
   </params>
</methodResponse>
```

CreateTemplate

XML Post

```
<?xml version="1.0"?>
<methodCall>
   <methodName>CreateTemplate</methodName>
   <params>
   <param>
<value>
  <struct>
    <member>
      <name>TemplateStyle</name>
      <value><string>5</string></value>
    </member>
    <member>
      <name>BBXMLRPCKey</name>
      <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
    </member>
  </struct>
</value>

   </param>
   </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
   <params>
   <param>
<value><boolean>1</boolean></value>
   </param>
   </params>
</methodResponse>
```

ReviseTemplate

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>ReviseTemplate</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>FocusHandle</name>
   <value><string>565745</string></value>
  </member>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

ManageTemplates

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>ManageTemplates</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
```

```
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

ReviseAssignments

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>ReviseAssignments</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>Index</name>
   <value><string>0</string></value>
  </member>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Fingerprint

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Fingerprint</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Template

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>Template</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>Index</name>
   <value><string>1</string></value>
  </member>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
```

```
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

ReviseUserData

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>ReviseUserData</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>Post Code</name>
   <value><string>01234</string></value>
  </member>
  <member>
   <name>Country</name>
   <value><string>USA</string></value>
  </member>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>

  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

FlipBadge

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>FlipBadge</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>ShowBack</name>
   <value><string>yes</string></value>
  </member>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

TeamTracerConfig

XML Post

```
<?xml version="1.0"?>
<methodCall>
  <methodName>TeamTracerConfig</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
```

```
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

TeamTracer

XML Sent

```
<?xml version="1.0"?>
<methodCall>
  <methodName>TeamTracer</methodName>
  <params>
  <param>
<value>
 <struct>
  <member>
   <name>BBXMLRPCKey</name>
   <value><string>Qf564FS4Fl67Sqap3DD5t</string></value>
  </member>
 </struct>
</value>
  </param>
  </params>
</methodCall>
```

XML Return

```
<?xml version="1.0"?>
<methodResponse>
  <params>
  <param>
<value><boolean>1</boolean></value>
  </param>
  </params>
</methodResponse>
```

Appendix

Non-Tryout Features

Dossier
Magnetic Encoding
No Video Capture
No Database Creation
Skin Tone Control
Browse Only
Single Photo
Fingerprint Enrollment
BBDirect Double Sided
BBDirect Video
BBDirect Options
Visitor Only
Smart Cards
VisitorPrimary

Tryout Features

TWAIN Interface
Signature Capture
Reports
Photo Capture from Bitmap
Double-sided Badges
Verifier
Video for Windows Capture
TeamTracer

Glossary of Terms

Access Key

A prearranged set of characters that the client must include in any XML communication for the server to permit responses to be generated

API

Application Programmer's Interface

bb32.ini

The initialization file for BadgeBuilder®

Boolean

A variable having only a TRUE or FALSE value

Client

The end of a client-server link responsible for asking the questions and interpretingthe answers

DNS

Domain Name Server

IP

An address on the Internet, for example 123.143.12.11

localhost

A user-friendly name for the computer the client is running on

Parameter

A value passed to a program call

Port

A number which defines the routing of internet protocol messages to an application

Server

The end of a client-server link responsible for listening for a question and providing the answer

Index

O

P

R

S

T

U

V

X