# BadgeBuilder® Integration Application Programmer's Interface

# Table of Contents

# Introduction

This document provides the application programmer with technical information detailing how to interface with BadgeBuilder®  to control its actions and obtain data from it.  It also describes the minimum requirements of a table in a database to allow BadgeBuilder®  to attach to it as a "foreign" source of information.

Generally the activities falls into two areas.  The first deals with how to control BadgeBuilder® to make it perform operations (like printing a badge) or return information (like the details of the currently selected badge).  The second deals with how BadgeBuilder® can share a table in a database with another application, such as a building access control system.  This sharing of information is important as it avoids the duplication of personnel data when it is required by both applications.

This document provides technical details of these interfaces.  It assumes the reader is already familiar with the technology involved in each case.

# Levels of Integration

BadgeBuilder® can be integrated with another application at several different levels. These are described briefly here in order of increasing complexity.

*Database Attachment Only*

In this scenario BadgeBuilder® is given access to the appropriate table in the application's database and all user operations are performed directly in BadgeBuilder®. There are risks that data may be changed by the user within BadgeBuilder® although these can be minimized by controlling access to BadgeBuilder® using its multi-level security system.

*Database Attachment and BadgeBuilder® Launch by the Application*

This method improves on the previous in that the application provides a button or other mechanism for launching BadgeBuilder®. The launching code can specify the name of the associated BadgeBuilder® file to open, this file having been pre-prepared by the developer. The launch code can also place BadgeBuilder® in *Server* mode as part of the launch process, providing a greater degree of data integrity protection.

*Dynamic Data Exchange (DDE)*

This well-established protocol for communication between applications can be used to provide a wide range of BadgeBuilder® functionality within your application. Methods of operation can be employed which make much of the day-to-day functionality of BadgeBuilder® invisible to the user. BadgeBuilder® can be instructed to pop up windows for badge preview and printing without showing the main BadgeBuilder® application window. This can result in a seamless and professional integration of badging with your application.

*Object Linking and Embedding (OLE)*

[Note: OLE is currently under development.]

This more recent method of communications between applications provides all the functionality previously described under *Dynamic Data Exchange***.**

# Foreign Database Requirements

## *Attaching to an Existing Database*

**Note:** not available in all versions of BadgeBuilder®.

BadgeBuilder® uses Open Database Connectivity (ODBC) to interface to the database. Therefore virtually any database type can be used if it included an ODBC driver for Microsoft Windows®. Database types know to work correctly include Microsoft Access, Paradox, dBase, FoxPro, IBM DB2, Microsoft SQL Server and PostgreSQL. Internally, the table must contain at least 5 fields. This is because every BadgeBuilder® table contains at least a key field, a field for the associated photograph file name, a field for the **Changed** flag and a template number field. Databases also must contain at least 1 user field, arriving at the numbers just mentioned.

You can connect to a pre-existing foreign database assuming that the database is of a type supported by the program. It must fulfill the following requirements:

1. It must contain a numeric field for the key. The data must be unique although BadgeBuilder® does not enforce this. Performance is improved if this field is indexed.
2. It must contain an alphanumeric field which must be capable of storing at least 12 characters for the photo file name.
3. It must contain a numeric field for the badge template number. This field should default to zero.
4. It must contain an alphanumeric field for the badge **Changed** flag.
5. It must contain at least one other numeric, date, time, timestamp or alphanumeric field to be used by BadgeBuilder®. You can allow BadgeBuilder® to use from 1 to 10 ( or an unlimited number if the Dossier option is activated) of these fields. If the foreign database contains more fields than BadgeBuilder® is attached to, then record deletion within BadgeBuilder® will not be allowed in any mode of operation.

For instructions on how to attach BadgeBuilder® to the a foreign table please see the BadgeBuilder® manual in the Design section, where the procedure is described in full.

## *Maintaining Database Integrity*

When BadgeBuilder® is attached to a foreign database the preferred method of operation is to allow the owning application to be responsible for the maintenance of all the data and BadgeBuilder® to be responsible for the badge design and generation.

As just noted, if BadgeBuilder® is attached to a table but is not associated with all the fields in the table, then deletion of a record by BadgeBuilder® is never allowed. BadgeBuilder® contains several general levels of security which are designed to control who can change information. In *Supervisory* or *Master* mode BadgeBuilder® will be able to modify existing records and add new records. However, an additional mode exists called *Server* which is identical to *Browse* mode (see manual), except that photo capture and badge printing is allowed and changing the user name is not. *Browse* mode, by definition, does not allow the BadgeBuilder® user to change any information, however a configuration setting in BadgeBuilder® will allow *Server* mode to change the current record if so desired.

# Dynamic Data Exchange (DDE) Interface

## *Specification*

This document provides information on the Dynamic Data Exchange (DDE) interface within BadgeBuilder®. This interface provides the application programmer with the ability to control certain operations within BadgeBuilder® from another application. These operations include opening and closing a database; locating and reading database records; adding, modifying and deleting records and terminating the program.

It is assumed that the reader is familiar with the concepts and principles behind DDE. No attempt is made here to describe the functionality of DDE.

**General**

BadgeBuilder® acts as a DDE Server only. This means that it responds to requests from a DDE Client who directs messages to it. The types of operations that can be requested of BadgeBuilder® fall into 3 general categories:

***Requests for Information (Request)***
The Client requests that BadgeBuilder® provide it with specific information. These requests include the data for the current badge (if any); the name of the database; and the status of the badge selection.

***Requests for Actions (Execute***)
In this type of operation the Client asks BadgeBuilder® to perform some specific action such as opening a database; selecting a specific record; deleting, changing or adding a record.

***Furnishing of Information (Poke)***
The Client uses this operation type when it wishes BadgeBuilder® to accept the information provided by the Client and use it for some future purpose. An example of this is when the Client wishes BadgeBuilder® to create a new badge record.

***Advise Link (Advise)***
The Client uses this operation to tell the BadgeBuilder® to advise the Client every time information of the requested type changes. This allows the BadgeBuilder® to advise the Client of the change without the client needing to continually issue requests. BadgeBuilder® supports both "hot" and "warm" advise links, although only the "warm" type has currently been tested.

**Starting BadgeBuilder®**

It is recommended that applications that wish to use BadgeBuilder®'s DDE interface, issue the [SERVERMODE] command immediately on establishing a DDE connection. This will allow BadgeBuilder® to operate in a special Server Mode, which is identical to Browse Mode (see manual), except that photo capture and badge printing is allowed and changing the user name is not. When BadgeBuilder® is in this operational mode, all access to other operational modes will be suspended, to maintain the integrity of the data. Also, when in Server Mode, any error message windows that would normally require a user interaction will not be displayed. However, please note that it is not required that the [SERVERMODE] command be issued to make use of the DDE interface.

**Communication Protocol**

Any DDE communications initiated by a Client require at least 2 vital items; the name of the intended server and the topic of interest within that server. For BadgeBuilder® the **server name** is **"BadgeBuilder®"** and this is true even when the product is given an OEM name other than BadgeBuilder®. The **topic name** determines in a general sense

the type of data to be manipulated. For example, this might categorize the data as related to a badge database or user information.

Additional information is usually required to complete a transaction and is dependent upon the type of transaction being performed. The following section describes in detail the transactions available and the form and nature of the data involved.

**Request Transactions**

This type of transaction is used by the Client to request specific information from BadgeBuilder®. The following **topic names** and transaction **item names** are supported:

***TOPIC NAME: "Database"***

**ITEM NAME: "Record"**

**Syntax 1: [Record]**

**Syntax 2: [Record <datakey value>]**

**DESCRIPTION: If a badge database is open and contains at least one badge, the database record is returned as a character string *(see appendix for data string formats).* If the database is not open the data is invalid and the error number is set. In syntax 1, the record returned is determined by the most recent POKE of a Database Record. Only the unique key part of the record is used, the remaining fields can be empty. In syntax 2, the key value can be included on the command line. In this case there is no requirement to previously poke a Database Record. In either case, if a record is found which matches the key, the data in the record is returned. Otherwise an error is returned and the error code set.**

**ITEM NAME: "Current"**

**DESCRIPTION: If a badge database is open and contains at least one selected badge, the database record related to the currently displayed badge is returned as a character string *(see appendix for data string formats).* If the database is not open the data is invalid and the error code is set.**

**ITEM NAME: "Count"**

**DESCRIPTION: The current count associated with the database is returned when a database is open. The values for the current record, number of selected records and total number of records is returned in a character string *(see appendix for data string formats)* . If the database is not open an error is returned and the error code set.**

**ITEM NAME: "Info"**

**DESCRIPTION: The database information associated with the database is returned when a database is open. The data is returned in a character string which contains the fully-defined file name; the underlying database manager type; the method of photograph compression and the related badge template *(see appendix for data string formats)* . If the database is not open an error is reported. This transaction can be used to determine if a database is open and if it is the desired one.**

**ITEM NAME: "Fields"**

**DESCRIPTION: The field names and related information associated with the database is returned when a database is open. The field names and data types are returned in a character**

**string *(see appendix for data string formats)*. If the database is not open an error is returned and the error code set.**

**ITEM NAME: "Templates"**

**DESCRIPTION: The number of templates assigned to the open database and their names. This record provides a means to discover how many templates (backdrops) are available for the current database. The data is returned in a character string *(see appendix for data string formats)*. All databases will have at least one template assigned to them.**

**ITEM NAME: "ExecutePending"**

**DESCRIPTION: This determines if an "Execute" command is pending or not. An "Execute" command is stored until the program returns to the main message loop (ie: the main screen) when it is then executed. The data is returned in a character string *(see appendix for data string formats)*.**

**ITEM NAME: "ToolbarStatus"**

**DESCRIPTION: This command determines the status of all the toolbar buttons at the actual time it is issued. It can be used, for example, to determine precisely which buttons are active when a particular badge is being displayed. Unlike the "Program" Item Name "Features", this determines not only if a feature is enabled but whether or not it is enabled for the particular badge or mode of operation.**

**It should be noted that only the status of the toolbar is returned, not the menu. If a toolbar button is not being displayed then it is considered unavailable. Therefore, this request might report a toolbar button not being available, because it is not showing, when the equivalent menu item is available.**

**The data is returned in a character string *(see appendix for data string formats)*.**

**ITEM NAME: "Handle"**

**DESCRIPTION: The handle of the BadgeBuilder® main window is returned. The data is returned in a character string *(see appendix for data string formats)*.**

**ITEM NAME: "Error"**

**DESCRIPTION: The error number and English text message of the last error incurred by the DDE interface. The data also contains an indication of whether or not an "Execute Transaction" is still pending. The data is returned in a character string *(see appendix for data string formats)*.**

**ITEM NAME: "EditCount"**

**DESCRIPTION: The number of edit windows in progress and not yet completed,as created by [EDIT], [CREATETEMPLATE] and [REVISETEMPLATE] commands *(see appendix for data string formats)*.**

***TOPIC NAME: "UserData"***

**ITEM NAME: "UserData"**

**DESCRIPTION: The user information record is returned as character string *(see appendix for character string formats).* This will return the contents of the structure whether or not a badge database is open. In BadgeBuilder® it is possible to enter User Information prior to opening a**

**database, although this information will be overwritten by any user information found in the database file.**

***TOPIC NAME: "Program"***

**ITEM NAME: "Features"**

**DESCRIPTION: The user information record is returned as character string *(see appendix for character string formats).* This will return the status of the optional features enabled. There are 64 possible features. The first 32 are either enabled or not. The second 32 are always available immediately after the program has been installed and can be used up to 20 times in aggregate. If they have not be purchased then the particular feature will become unavailable after that. The returned data will show the current enable status for all 64 features.**

**ITEM NAME: "Busy"**

**DESCRIPTION: Returns information on the state of BadgeBuilder®. BadgeBuilder® is busy when a command is in operation and is idle when the command is completed. BadgeBuilder® remains busy whenever a user dialog is open and returns to idle when the dialog is closed. *(see appendix for data string formats)*.**

**ITEM NAME: "Info"**

**DESCRIPTION: The database information associated with the database is returned when a database is open. The data is returned in a character string which contains the fully-defined file name; the underlying database manager type; the method of photograph compression and the related badge template *(see appendix for data string formats)* . If the database is not open then those fields related to a database will be blank. This transaction can be used to determine if a database is open and if it is the desired one.**

**Execute Transactions**

Execute transactions are sent to the server to tell it to take some kind of action. The server tries to perform the action and reports the results of its efforts to the Client. The following **topic names** and transaction **command strings** are supported. Command strings are not case sensitive; arguments must be separated by at least one space; and the *'['* and *']'* are required. Arguments are shown in the syntax in angle brackets to indicate a substitution is required.

**NOTE:** Execute Transactions only get acted upon while BadgeBuilder® is in its main idle loop (ie: no user operation is in progress and no modal dialogs are pending). Requests for transaction execution are accepted immediately and stored for future execution. This allows requests to be sent without regard to the current operation in progress within BadgeBuilder®. As a result, Execute Transactions will always return successfully even though the pending action may not ultimately be successful. (One exception to this is that an Execute Transaction will fail if a previous Execute Transaction is pending and has not been performed. The existence of a pending transaction can be determined by the Request Transaction item name "ExecutePending".) The success or failure of an executed transaction can be determined by the Request Transaction item name "Error".

***TOPIC NAME: "Database"***

**COMMAND: "Open"**

**SYNTAX: [OPEN <filename>]**

**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 <filename> must be a fully qualified BadgeBuilder® database filename with the *".IDC"* extension. Failure to open the database is reported through the normal DDE error response for Execute Transactions. 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.**

**COMMAND: "Close"**

**SYNTAX: [CLOSE]**

**DESCRIPTION: This command close a database which is currently open if it exists, otherwise it will do nothing. No error is returned if a database is not open.**

**COMMAND: "First"**

**SYNTAX: [FIRST]**

**DESCRIPTION: Select the first record from the current selection in the database. This will return an error if the database is not open or contains no records.**

**COMMAND: "Next"**

**SYNTAX: [NEXT]**

**DESCRIPTION: Select the next record from the current selection in the database. This will return an error if the database is not open or contains no records.**

**COMMAND: “Previous”**

**SYNTAX: [PREVIOUS]**

**DESCRIPTION: Select the previous record from the current selection in the database. This will return an error if the database is not open or contains no records.**

**COMMAND: “Last”**

**SYNTAX: [LAST]**

**DESCRIPTION: Select the last record from the current selection in the database. This will return an error if the database is not open or contains no records.**

**COMMAND: “GoTo”**

**SYNTAX: [GOTO n]**

**DESCRIPTION: Select the nth record from the current selection in the database. This will return an error if the database is not open, contains no records, or the value of “n” is outside the range of badges currently selected. Note that the first badge is numbered “1”.**

**COMMAND: “Select Where”**

**SYNTAX 1: [SELECT WHERE <criteria>]**

**SYNTAX 2: [SELECT WHERE]**

**DESCRIPTION: Select the records in the database which match the given criteria. This will return an error if the database is not open. In syntax 1, <criteria> is a standard SQL predicate. Some of the more common and useful ones are:**

**comparison predicate ( = ,<, <=, >, >=, <> )**

e.g.: SELECT WHERE NUMBER < 10

All records with the numeric field less than 10 will be selected

**between predicate**

e.g.: SELECT WHERE NAME BETWEEN 'BROWN' AND 'BRUTUS'

Selects all records having character strings in the NAME field between 'BROWN' & 'BRUTUS', inclusively

**in predicate**

e.g.: SELECT WHERE NAME IN ('FRED','JOHN','BILL')

Only records have a name in the NAME field equal to one of the 3 names will be selected

**like predicate**

e.g.: SELECT WHERE NAME LIKE 'ARCH%'

All records where the NAME field starts with 'ARCH' will be included

**Predicates can be joined with logical** AND **and** OR**, and** NOT **can be used to reverse the logical action. As a result**: SELECT WHERE NAME LIKE 'ARCH' AND NOT NUMBER < 10 **selects all records where the NAME field starts with 'ARCH' and the NUMBER field is 10 or greater.**

**Character comparisons are collated according the ASCII sequence and as a result ARE case-sensitive. In the LIKE predicate, a '%' represents zero or more characters and '_' represents any single character.**

**For more information on SQL predicates and their usage see "SQL Instant Reference" by Martin Gruber, published by Sybex.]**

**In syntax 2, all records currently in the database will be selected.**

**In either case, the Request Transaction "Count" can be used to determine the results of the select command.**

**COMMAND: "Delete"**

**SYNTAX 1: [DELETE]**

**SYNTAX 2: [DELETE <datakey value>]**

**DESCRIPTION: Delete a record from the database. This will return an error if the database is not open or contains no records. On syntax 1, the record deleted is the one with a key matching the key in the record last sent by the POKE command. If no record has been poked yet and error will result. In syntax 2, the key value is included with the command. In either case, if no key match is found nothing happens.**

**COMMAND: "Revise"**

**SYNTAX: [Revise]**

**DESCRIPTION: Revise a record in the database according to the data last sent by the POKE command (see below). This will return an error if the database is not open, contains no records, or there is no key match with the data most recently supplied by the POKE command.**

**This will revise the matching record according to the data in the record data last POKED by the Client. Changing the contents of a record is a 2 step process:**

1. **POKE the record data with required changes**
2. **Issue the REVISE command**

**COMMAND: "Add"**

**SYNTAX: [ADD]**

**DESCRIPTION: Add a record to the database. This will return an error if the database is not open, already contains a record with the same key or contains no records. This will add a record containing the data in the record character string last POKED by the Client. If an ADD command is used without first POKING a new record, an error will result. Adding a record is a 2-step process:**

1. **POKE the record data with an unused key**
2. **Issue the ADD command**

**COMMAND: "Update"**

**SYNTAX: [UPDATE]**

**DESCRIPTION: Change an existing or add a new record to the database. This will return an error if the database is not open. This will add a record containing the data in the record character string last POKED by the Client if a record with a matching key is not found. If a key match is found, the matching record is updated. If an UPDATE command is used without first POKING a record an error will result. Updating a record is a 2-step process:**

1. **POKE the record data with a used or unused key**
2. **Issue the UPDATE command**

**COMMAND: "Exit"**

**SYNTAX: [EXIT]**

**DESCRIPTION: Close the current database, if any, and terminates BadgeBuilder®.**

**COMMAND: "StartLog"**

**SYNTAX: [STARTLOG]**

**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. (The format and content of this file is defined in Appendix II.)**

**COMMAND: "StopLog"**

**SYNTAX: [STOPLOG]**

**DESCRIPTION: Turns BadgeBuilder® event logger off.**

**COMMAND: "Restore"**

**SYNTAX: [RESTORE]**

**DESCRIPTION: Restores the BadgeBuilder® window and gives it the focus. This will restore it from either the maximized or minimized state. No error results.**

**COMMAND: "DDEMODE"**

**SYNTAX 1: [DDEMODE ON]**

**SYNTAX 2: [DDEMODE OFF]**

**DESCRIPTION: Syntax 1 places BadgeBuilder® in the Server Mode. Syntax 2 returns it to Browse Mode. An error results from either syntax if BadgeBuilder® is in Demo Mode. Server mode is described above.**

**[Note: This command is deprecated in favor of the [SERVERMODE] command, which can exactly duplicate its function. New applications should not use this command.]**

**COMMAND: "SERVERMODE"**

**SYNTAX 1: [SERVERMODE ON]**

**SYNTAX 2: [SERVERMODE OFF]**

**SYNTAX 3: [SERVERMODE n]**

**DESCRIPTION: Syntax 1 places BadgeBuilder® in the Server Mode. Syntax 2 returns it to Browse Mode. Syntax 3 places BadgeBuilder® in Server Mode and also modifies the BadgeBuilder® window as detailed below. An error results from any syntax if BadgeBuilder® is in Demo Mode. Server mode is described above.**

**When Syntax 3 is used BadgeBuilder® takes the following actions:**

1. Removes the menu and status bar
2. Modifies the toolbar according to the value of "n" (see below)
3. Prevents the user from terminating BadgeBuilder® from the "Close" command
4. If a database is open, reduce the window size to encompass the current badge while still keeping the entire toolbar visible. The Window size will automatically adjust as different badge shapes are displayed.

**The value "n" is a binary number where 1 indicates that the related toolbar button is to be displayed, and "0" not. The least significant (i.e. right-most) digit in "n" relates to the "Database Open" toolbar button. The order of the remaining buttons is as shown in the appendix for the [TOOLBARSTATUS] request returned data.**

**NOTE**

**Although you can use the toolbar buttons displayed in BadgeBuilder® as an aid in determining their position in the number, you will need to allow for the fact that not all buttons are necessarily being displayed. For example, in BadgeBuilder® the "PHOTO FROM BITMAP" is not shown.**

**A special case exists when "n" is given the value zero. A default toolbar will be displayed containing the Badge Side Selection, Video Capture, TWAIN Capture, Bitmap Capture, Signature Capture, Fingerprint Capture, Print and Help buttons. This [SERVERMODE] command is equivalent to the command**

**[SERVERMODE 110011111100000000000000]**

**Once a Syntax 3 command has been issued, another cannot be issued without an intervening command using the Syntax 1 or 2 form.**

**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.**

**COMMAND: "VIDEO"**

**SYNTAX: [VIDEO]**

**DESCRIPTION: This command causes BadgeBuilder® to open its video capture window the next time it returns to the main idle loop (see above). It will only do so if the "Video" toolbar button is available and sending this command is equivalent to clicking on this tool button. An error will return if no database is open, no video capture board is installed or the currently selected badge does not include a photo holder rectangle. 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.**

**COMMAND: "TWAIN"**

**SYNTAX: [TWAIN]**

**DESCRIPTION: This command causes BadgeBuilder® to open its TWAIN capture window the next time it returns to the main idle loop (see above). It will only do so if the "TWAIN" toolbar button is available and sending this command is equivalent to clicking on this tool button. An error will return if no database is open, no TWAIN sources are installed or the currently selected badge does not include a photo holder rectangle. 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.**

**COMMAND: "Bitmap"**

**SYNTAX: [Bitmap]**

**DESCRIPTION: This command causes BadgeBuilder® to open its Bitmap capture window the next time it returns to the main idle loop (see above). It will only do so if the "Bitmap" listing is available under the "Photo" heading. Sending this command is equivalent to clicking on this listing. An error will return if no database is open, or the currently selected badge does not include a photo holder rectangle. 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.**

**COMMAND: "Signature"**

**SYNTAX: [Signature]**

**DESCRIPTION: This command causes BadgeBuilder® to open its Signature capture window the next time it returns to the main idle loop (see above). It will only do so if the "Signature" toolbar button is available and sending this command is equivalent to clicking on this tool button. An error will return if no database is open, or the currently selected badge does not include a signature holder rectangle. 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.**

**COMMAND: "FOCUS"**

**SYNTAX 1: [FOCUS n]**

**SYNTAX 2: [FOCUS]**

**DESCRIPTION: This command causes BadgeBuilder® to switch the window focus whenever any other DDE command is completed and the main idle loop is again entered. In syntax 1, "n" is the decimal value if the window handle to receive the focus when the command has completed executing. Note that no focus change occurs when this command is issued. Rather it determines what focus action should be taken when other "Execute" commands are completed. Syntax 2 disables the focus change mechanism and is equivalent to sending the value 0 for "n" is syntax 1.**

**COMMAND: "PRINT"**

**SYNTAX: [PRINT]**

**DESCRIPTION: This command causes BadgeBuilder® to open its printing dialog window the next time it returns to the main idle loop (see above). It will only do so if the "Printer" toolbar button is available and sending this command is equivalent to clicking on this tool button. An error will return if no database is open. 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.**

**COMMAND: “Preview”**

**SYNTAX: [Preview]**

**DESCRIPTION: This command displays a preview of the badge. A window appears as shown below. 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.**


**COMMAND: “Edit”**

**SYNTAX: [EDIT m n]**

**DESCRIPTION: This command causes the BadgeBuilder® window to focus and execute a “Revise Current Badge Design” 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 edituser operation is completed, as long as [SERVERMODE] is not OFF.**

**It is recommended that syntax 3 of [SERVERMODE is used to control which main toolbar icons are displayed, to hide the menu system and prevent the user from closing BadgeBuilder®.**

**This command will allow editing of the badge appearance for the currently selected badge. It is possible to issue this command multiple times for either the same or different badges which use the same or different templates. However, if a request to edit the same template occurs, only one edit session will start. It is recommended for operational clarity that only one command is issued at a time. The number of currently active [EDIT] sessions can be determined by issuing an [EDITCOUNT] request.**

**If “m” is specified, it is the template number in the current database which to edit. To edit the template associated with the current badge either leave “m” out or set it to –1.**

**If “n” is specified it should be the handle of the window to receive the focus when the edit process is completed by the user. This window handle must be in decimal. To specify “n” you must specify “m”.**

**COMMAND: "ReviseTemplate"**

**SYNTAX: [ReviseTemplate n]**

**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.**

**It is recommended that syntax 3 of [SERVERMODE is used to control which main toolbar icons are displayed, to hide the menu system and prevent the user from closing BadgeBuilder®.**

**This command will allow the user to select an existing template (.ctm) file for editing. It is possible to issue this command multiple times. However, it is recommended for operational clarity that only one command is issued at a time. The number of currently active template edit sessions can be determined by issuing an [EDITCOUNT] request.**

**If "n" is specified it should be the handle of the window to receive the focus when the edit process is completed by the user. This window handle must be in decimal.**

**COMMAND: "CreateTemplate"**

**SYNTAX: [CreateTemplate m n]**

**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 not OFF.**

**It is recommended that syntax 3 of [SERVERMODE is used to control which main toolbar icons are displayed, to hide the menu system and prevent the user from closing BadgeBuilder®.**

**This command will allow the creation of a new badge design template file (.ctm file) which can be imported into a database using the [ManageTemplates] command. It is possible to issue this command multiple times. However, it is recommended for operational clarity that only one command is issued at a time. The number of currently active template edit sessions can be determined by issuing an [EDITCOUNT] request.**

**If "m" is specified, it is the template size to create. If "m" is not specified, then type 6 will be assumed The types are:**

| | |
|---|---|
| **VERTICAL CREDIT** | **0** |
| **HORIZONTAL CREDIT** | **1** |
| **VERTICAL IBM** | **2** |
| **HORIZONTAL IBM** | **3** |
| **VERTICAL ALLPHOTO** | **4** |
| **HORIZONTAL ALLPHOTO** | **5** |
| **USERCARDSIZE** | **6** |

**If "n" is specified it should be the handle of the window to receive the focus when the edit process is completed by the user. This window handle must be in decimal. To specify "n" you must specify "m".**

**COMMAND: “ManageTemplates”**

**SYNTAX: [ManageTemplates]**

**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.**

**This command will allow the user to manage the templates associated with the database, including adding new templates from .ctm template files, over-riding existing templates, deleting specific unused templates and purging all unused templates.**

**This is a modal command, so it can be preceeded by a [FOCUS] command which specifies the handle of the window to receive the focus when the command is complete.**

**COMMAND: “ReviseAssignments”**

**SYNTAX: [ReviseAssignments n]**

**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.**

**The template number "n" is optional. If not given, the field assignment of the currently selected badge will be used. Otherwise, this is a zero-based number indicating the template to have its fields reassigned. If "n" is invalid an "Invalid Template Number" will be returned.**

**This is a modal command, so it can be preceeded by a [FOCUS] command which specifies the handle of the window to receive the focus when the command is complete.**

**COMMAND: “Template”**

**SYNTAX: [Template n]**

**DESCRIPTION: This command causes the currently selected badge to have its template set to the given number. The new appearance of the badge will also display immediately. If "n" is invalid an "Invalid Template Number" will be returned. The database must be open for this command to function.**

***TOPIC NAME: “UserData”***

**COMMAND: “Revise”**

**SYNTAX: [REVISE]**

**DESCRIPTION Revise the user data. This will change the user data using the data in the record character string last POKED by the Client. If a REVISEcommand is used without first POKING the data, an error will result. Revising the data is a 2-step process:**

3. **POKE the record data**
4. **Issue the REVISEcommand**

## Poke Transactions

Poke transactions are used to send data to the server. The server stores the data and does nothing with it until an EXECUTE TRANSACTION instructs it to do so.

A mechanism is provided to allow the DDE client to specify if data in any field is to be left unchanged. If the field is an alphanumeric type, then providing a null string for that field will leave the existing data unchanged. To clear a field, place a single space character in the associated poke field. The field in the database will be cleared. For numeric fields, the existing data can be left unchanged by making the associated field in the poked data a null string. For numeric fields, there is no concept of an empty field in the database. A value is always present. The ADD, UPDATE or REVISE command can never change counter fields, so any value can be put in the associated field.

The following **topic names** and transaction **item names** are supported.

***TOPIC NAME: "Database"***

**ITEM NAME: "Record"**

**DESCRIPTION: The data contained in the character data string is stored to await further instruction. If the database is not open, an error is returned. The exchange data format used is 'CF_TEXT'. The data is an array of text characters. Each line ends with a carriage return–linefeed (CR-LF) combination. A null character signals the end of the data.** ***(See the Windows 3.1 SDK for more information.)***

**The data stored is used in conjunction with the topic *'Database'* commands ADD, UPDATE and REVISE.**

***TOPIC NAME: "UserData"***

**ITEM NAME: "Record"**

**DESCRIPTION: The data contained in the character data string is stored to await instruction. The exchange data format used is 'CF_TEXT'. The data is an array of text characters. Each line ends with a carriage return–line feed (CR-LF) combination. A null character signals the end of the data.** ***(See the Windows 3.1 SDK for more information.)***

**The data stored is used in conjunction with the topic *'UserData'* command REVISE.**

## Advise Link

The Client sets up an Advise Link with BadgeBuilder® for a given topic and item name. Once established, BadgeBuilder® will advise the Client whenever the associated data changes. BadgeBuilder® supports both "hot" and "warm" advise links.

***TOPIC NAME: "Program"***

**ITEM NAME: "Busy"**

**DESCRIPTION: Returns information on the state of BadgeBuilder®. BadgeBuilder® is busy when a command is in operation and is idle when the command is completed. BadgeBuilder® remains busy whenever a user dialog is open and returns to idle when the dialog is closed.** ***(see appendix for data string formats).***

## Data Record

***Data Record Formats***

Data records are transmitted using text characters in a one-string record. Each field of the record is separated with a carriage-return/line-feed character pair and the entire string is NULL terminated. The number of fields is fixed unless noted otherwise. For fixed field-count records, all fields are present regardless of the number of fields in use. All fields are alphanumeric strings of arbitrary length but limited by the type of data represented. The first field is always a unique number indicating the type of data field involved.

***TOPIC NAME: "Database"***

**ITEM NAME: "Record"**

**DESCRIPTION: This record contains the variable data for one badge. It contains a unique record key number, the name of the photograph file, the index number of the associated template (backdrop) and the string data for the 10 fields (an unlimited number if the Dossier Feature is enabled).**

**The type of field (alphanumeric or numeric) is determined during the construction of the BadgeBuilder® database. A database can have up to 10 fields (an unlimited number if the Dossier Feature is enabled) in any mix of numeric and alphanumeric types. Undefined fields will have zero length strings at that location in the record. Defined fields are contiguous starting at "field 1".**

**The path to the photograph file can be determined by requesting the data record for the item name *'Info'*. The *'Field'* record provides information on whether the alphanumeric of numeric field is active for each field, and for the latter, how many decimal places are used.**

**A decompression routine is available from Ashdown for the RLE method (which, by the way, is not really RLE but is actually based on "Compress and decompress files using the "splay tree" technique. Based on an article by Douglas W. Jones, "Application of Splay Trees to Data Compression", in Communications of the ACM, August 1988, page 996). A JPEG compression/decompression 'C' library is available on CompuServe from The Independent JPEG Group. The ZIP file is available from Ashdown. Fractal files can only be decompressed using code available only from Iterated Systems, Inc.**

**DETAIL: The record format is as follows:**

<record id><cr/lf>
<unique key number><cr/lf>
<photograph file name> <cr/lf>
<template index number><cr/lf>
<change flag><cr/lf>
<field 1 > <cr/lf>
<field 2 > <cr/lf>
<field 3 > <cr/lf>
<field 4 > <cr/lf>
<field 5 > <cr/lf>
<field 6 > <cr/lf>
<field 7 > <cr/lf>
<field 8 > <cr/lf>
<field 9 > <cr/lf>
<field 10 > <cr/lf>
[<field 11 > <cr/lf>

.
.
<field 50 > <cr/lf>
.
.
]
NULL

***Specific information on each field is:***

| | |
|---|---|
| <record id> | **contains "1" to indicate the record type** |
| <unique key number> | **a number which uniquely identifies the record. It can be any value based upon a 'C' long data type. The key value is required.** |
| <photograph file> | **contains the name and extension only. This field is read-only.** |
| <template index numeric> | **The index number of the badge template to use with this record. This is the index number, zero-based of the template to use. If outside the acceptable range, zero will be used and an error generated. There is always at least one template.** |
| <change flag> | **This is an empty string if the badge data has not changed since it was last printed. If it has changed, the string will not be empty (normally it will contain an "X").** |
| <field x > | **If the field type is alphanumeric, it can be up to 40 characters long. If the field is numeric or counter, it is based on the 'C' double data type.** |

**All 10 fields (50 or more if the Dossier Feature option is enabled) for each type are always present (although the strings may be of zero length) regardless of how many fields are active in the database or which type are in use. To maintain compatibility with earlier versions of the program, if the Dossier Feature is enabled, at least 50 fields will always be output, even if less than that are used. If more than that are used, then the exact number are output. All fields contain ASCII representations of the data, even the key and numeric fields.**

**ITEM NAME: "Count"**

**DESCRIPTION: This record contains three integer numbers which represent the index of the currently selected badge, the number of badges currently selected and the total number of badges in the database.**

**DETAIL: The record format is as follows:**

<record id><cr/lf>
<badge index number> <cr/lf>
<selected badge count> <cr/lf>
<total badge count> <cr/lf>
NULL

***Specific information on each field is:***

| | |
|---|---|
| <record id> | **contains "2" to indicate the record type** |
| <badge index number> | **Current badge in selection** |
| <selected badge count> | **Number of records selected** |
| <total badge count> | **Total number of records** |

The values are all based in the 'C' long data type and will never be negative. The badge index number is 1 based.

**ITEM NAME: "Info"**

**DESCRIPTION: This record contains information related to the database.**

**DETAIL: The record format is as follows:**

<record id><cr/lf>
<database filename> <cr/lf>
<database manager type> <cr/lf>
<photo compression method> <cr/lf>
<photograph path> <cr/lf>
<dossier template index> <cr/lf>
<program version number> <cr/lf>
<product version number> <cr/lf>
<program mode><cr/lf>
<tryouts left><cr/lf>
NULL

***Specific information on each field is:***

| | |
|---|---|
| <record id> | **contains "3" to indicate the record type** |
| <database filename> | **The full filename of the IDC file currently open** |
| <database manager type> | **The database type (Paradox, Access, etc.)** |
| <photo compression method> | **A number representing the compression scheme as follows: 0=Uncompressed PCX, 1=RLE Compressed PCX, 2=JPEG, 3=Fractal** |
| <photograph path> | **The path to the photograph files. This includes the trailing slash.** |
| <dossier template index> | **The index number of the dossier template in use, or –1 if none defined** |
| <program version number> | **The version number of the current copy of BadgeBuilder® running, for example 3.3.2** |
| <product version number> | **The product version number of the current copy of BadgeBuilder® running as 4 comma-separated digits, eg: 3,3,2,0** |
| <program mode> | **0=Browse, 1=Normal, 2=Supervisory, 3=Master, 4=Demo, 5=DDE** |
| <tryouts left> | **Returns the number of try outs left in the security key, or zero if no key is present.** |

**ITEM NAME: “Fields”**

**DESCRIPTION: This record contains information related to the fields in the database. This record is read-only. All 10 fields (50 or more if the Dossier Feature option is enabled) for each type are always present (although the strings may be of zero length) regardless of how many fields are active in the database or which type are in use. To maintain compatibility with earlier versions of the program, if the Dossier Feature is enabled, at least 50 fields will always be output, even if less than that are used. If more than that are used, then the exact number are output.**

**DETAIL: The record format is as follows:**

<record id><cr/lf>
<unique key number><cr/lf>
<photograph file name> <cr/lf>
<template index number><cr/lf>
<change flag><cr/lf>
<active field count> <cr/lf>
<field1 name> <cr/lf>
<field2 name> <cr/lf>
<field3 name> <cr/lf>
<field4 name> <cr/lf>
<field5 name> <cr/lf>
<field6 name> <cr/lf>
<field7 name> <cr/lf>
<field8 name> <cr/lf>
<field9 name> <cr/lf>
<field10 name> <cr/lf>
[<field11 name > <cr/lf>
.
.
<field50 name > <cr/lf>
.
.
]
<field1 type> <cr/lf>
<field2 type> <cr/lf>
<field3 type> <cr/lf>
<field4 type> <cr/lf>
<field5 type> <cr/lf>
<field6 type> <cr/lf>
<field7 type> <cr/lf>
<field8 type> <cr/lf>
<field9 type> <cr/lf>
<field10 type> <cr/lf>
[<field11 type > <cr/lf>
.
.
<field50 type > <cr/lf>
.
.
]
<field1 decimal places> <cr/lf>
<field2 decimal places> <cr/lf>

<field3 decimal places> <cr/lf>
<field4 decimal places> <cr/lf>
<field5 decimal places> <cr/lf>
<field6 decimal places> <cr/lf>
<field7 decimal places> <cr/lf>
<field8 decimal places> <cr/lf>
<field9 decimal places> <cr/lf>
<field10 decimal places> <cr/lf>
[<field11 decimal places > <cr/lf>
.
.
<field50 decimal places > <cr/lf>
.
.
]
<field1 alias name> <cr/lf>
<field2 alias name> <cr/lf>
<field3 alias name> <cr/lf>
<field4 alias name> <cr/lf>
<field5 alias name> <cr/lf>
<field6 alias name> <cr/lf>
<field7 alias name> <cr/lf>
<field8 alias name> <cr/lf>
<field9 alias name> <cr/lf>
<field10 alias name> <cr/lf>
[<field11 alias name> <cr/lf>
.
.
<field50 alias name> <cr/lf>
.
.
]
NULL

***Specific information on each field is:***

| | |
|---|---|
| <record id> | **contains "4" to indicate the record type** |
| <active field count> | **The number of fields used in the database, will always be 1 or higher** |
| <fieldx name> | **The name of each field** |
| <fieldx type> | **0 = alphanumeric, 1 = numeric, 2 = counter** |
| <fieldx alias name> | **field alias name or an empty string** |
| <fieldx decimal places> | **Number of decimal places, valid for type 1 above only, for type 2, decimal places is always zero** |

**ITEM NAME: "ExecutePending"**

**DESCRIPTION: This record a string indicating if there is an execute instruction which has not yet been carried out.**

**DETAIL: The record format is as follows:**

<record id><cr/lf>
<bool value><cr/lf
<pending text><cr/lf
<inmainloop text><cr/lf

NULL

***Specific information on each field is:***

| | |
|---|---|
| <record id> | **contains "5" to indicate the record type** |
| <bool value> | **0 for False, 1 from True** |
| <pending text> | **Contains "TRUE" if an execute transaction is still pending or "FALSE" if not.** |
| <inmainloop text> | **Contains "TRUE" if BadgeBuilder® is in the main idle loop or "FALSE" if not. This can be used to detect when a BadgeBuilder® dialog is open.** |

**ITEM NAME: "Error"**

**DESCRIPTION: This record contains the number and text of the last DDE error encountered.**

**DETAIL: The record format is as follows:**

<record id><cr/lf>
<error number> <cr/lf>
<error text> <cr/lf>
<pending text><cr/lf
<inmainloop text><cr/lf
NULL

***Specific information on each field is:***

<record id> **contains "6" to indicate the record type**

| <error number> | <error text> |
|---|---|
| 0 | No Error |
| 1 | Database not open |
| 2 | Record not found |
| 3 | Non-numeric data in a numeric field |
| 4 | No data for execution command |
| 5 | Cannot find database |
| 6 | Unknown execute command |
| 7 | Unknown item name |
| 8 | Unknown topic name |
| 9 | Invalid poke field format |
| 10 | Too many characters in field data |
| 11 | Invalid template number |
| 12 | No index number |
| 13 | Command already pending |
| 14 | Invalid data type |
| 15 | No records selected |
| 16 | File not found |
| 17 | Already at first record |
| 18 | Already at last record |

| | |
|---|---|
| 19 | Record already exists |
| 20 | Insert failed |
| 21 | Key number outside range |
| 22 | DDE mode change failed |
| 23 | Unable to switch focus |
| 24 | Video capture not available |
| 25 | Printing not available |
| 26 | TWAIN capture not available |
| 27 | Signature capture not available |
| 28 | ODBC Data Source configuration error |
| 29 | Counter Field Overflowed Fixed Length |
| 30 | Bitmap Capture not available |
| 31 | Index outside valid range |
| 32 | Invalid template number |
| 33 | Fingerprint capture not available |
| <pending text> | Contains **"TRUE"** if an execute transaction is still pending or **"FALSE"** if not. |
| <inmainloop text> | **Contains "TRUE" if BadgeBuilder® is in the main idle loop or "FALSE" if not. This can be used to detect when a BadgeBuilder® dialog is open.** |

**ITEM NAME: "Templates"**

**DESCRIPTION: This record contains the number of templates (badge backdrops) available and the user name for each one. There will always be at least one template. Note that the record field count is variable.**

**DETAIL: The record format is as follows:**

<record id><cr/lf>
<number of templates> <cr/lf>
<template 0 name> <cr/lf>
.
.
.
<template 'n' name> <cr/lf>
NULL

***Specific information on each field is:***

| | |
|---|---|
| <record id> | **contains "7" to indicate the record type** |
| <number of templates> | **The number of defined templates in the database. This will always be one or greater.** |
| <template 'n' name> | **The user name for each template. This field will contain a string in the form "nn - aaaaaaaa", for example 2 - ACCOUNTS. The number "nn" can be any number of digits long, but in practice will normally be one or 2 digits. The string "aaaaaaaa" will be variable in length.** |

**ITEM NAME: “ToolbarStatus”**

**DESCRIPTION: This record contains the number of toolbar buttons available and their current status. TRUE is returned if the specific toolbar button is both displayed and active. As this status may change from badge to badge, the data is only valid until another badge is selected.**

**DETAIL: The record format is as follows:**

<record id><cr/lf>
<number of toolbar buttons> <cr/lf>
<toolbar button1 status> <cr/lf>
.
.
.
<toolbar button”n” status> <cr/lf>
NULL

***Specific information on each field is:***

<record id> **contains "10" to indicate the record type**

<number of toolbar buttons> **The number of toolbar buttons available.**

< toolbar button”n” status > **Either TRUE or FALSE, indicating the status of each button. The order of the toolbar buttons is as follows:**

DATABASE OPEN
CREATE NEW DATABASE
USER LOG IN
CREATE NEW BADGE
EDIT EXISTING BADGE
FIRST BADGE
PREVIOUS BADGE
NEXT BADGE
LAST BADGE
SELECT ALL BADGES
FAST FIND
SELECT
SORT
VERIFY
DOSSIER
FLIP BADGE
VIDEO CAPTURE
TWAIN ACQUIRE
PHOTO FROM BITMAP
CAPTURE SIGNATURE
CAPTURE FINGERPRINT
DELETE CURRENT BADGE
DELETE ALL SELECTED BADGES
PRINT
HELP

**ITEM NAME: "Handle"**

**DESCRIPTION: This record is a string indicating BadgeBuilder®'s main Window Handle.**

**DETAIL: The record format is as follows:**

<record id><cr/lf>
<handle><cr/lf
NULL

***Specific information on each field is:***

| | |
|---|---|
| <record id> | **contains "11" to indicate the record type** |
| <handle> | **windows handle in base 10** |

**ITEM NAME: "EditCount"**

**DESCRIPTION: This record is a string indicating the number of currently active [EDIT] sessions.**

**DETAIL: The record format is as follows:**

<record id><cr/lf>
<session count><cr/lf
NULL

***Specific information on each field is:***

| | |
|---|---|
| <record id> | **contains "12" to indicate the record type** |
| <session co> | **number of current sesssions** |

**ITEM NAME: "Busy"**

**DESCRIPTION: This record is a string indicating the busy status of BadgeBuilder®.**

**DETAIL: The record format is as follows:**

<record id><cr/lf>
<pending text> Contains "TRUE" if BadgeBuilder® is busy or "FALSE" if not.
NULL

***Specific information on each field is:***

| | |
|---|---|
| <record id> | **contains "11" to indicate the record type** |
| <handle> | **windows handle in base 10** |

## ***TOPIC NAME: "UserData"***

**ITEM NAME: "UserData"**

**DESCRIPTION: This record contains data stored as User Data in BadgeBuilder®.**

**DETAIL: The record format is as follows:**

<record id><cr/lf>
<company> <cr/lf>
<street> <cr/lf>
<city> <cr/lf>

<state> <cr/lf>
<zipcode> <cr/lf>
<country> <cr/lf>
NULL

**There is no limit on the length of the strings other than the practical one of the ability to display the data on the badge. The record id number is 8.**

## ***TOPIC NAME: "Program"***

**ITEM NAME: "Features"**

**DESCRIPTION: This record contains data regarding the status of optional features.**

**DETAIL: The record format is as follows:**

<record id><cr/lf>
<feature1><cr/lf>
.
.
.
<feature64><cr/lf>
NULL

***Specific information on each field is:***

<record id> **contains "9" to indicate the record type**

<feature1> - <feature64> **TRUE if enabled , FALSE otherwise**

The features are as follows:

| | |
|---|---|
| 1 | Dossier |
| 2 | Magnetic Encoding |
| 3 | No Video Capture |
| 4 | No New Database |
| 5 | Skin Tone Control |
| 6 | Browse Only |
| 7 | Single Photo |
| 8 | Fingerprint Analysis |
| 9 | LXI Double Sided |
| 10 | LXI Video Capture |
| 11 | LXI Options |
| 12-32 | Unassigned |
| 33 | TWAIN interface |
| 34 | Signature capture |

35 Reports

36 Bitmap Photos

37 Double Sided Badges

38 Verifier

39-64 Unassigned

# Object Linking and Embedding (OLE)

[Note: The OLE interface is currently under development. Functionally it will provide the same capabilities as DDE.]

# Event Logger

## *Event Logger Records*

When event logging is turned on, a file called **BB.LOG** is generated for each action taken by the program, whether it is via a DDE action or directly by the user. The log file will be placed in the same directory as the main **BB.EXE** program and will be created the first time a record is to be written, if it does not already exist. To avoid possible disk corruption the log file is opened and closed for each transaction written. It is **never** purged or deleted by BadgeBuilder®. This is the responsibility of the program or user making use of the log file. Failures of the program to open or write to the log file will be silently ignored (as could occur with a full disk).

Event logging can be initiated and terminated via the File menu and the DDE Execute Transaction command. It can also be initiated when BadgeBuilder® is launched by including the switch **/LOG** on the command line.

The following events are recorded into the log file:

- Adding a badge record
- Deleting a badge record
- Changing a badge record (including capturing a photograph, or changing a template)
- Opening a database
- Closing a database
- Changing the user data
- Print a badge or selection of badges
- Adding or deleting a database field
- Revising a existing template layout
- Adding or changing the template list
- Reassigning fields on a template
- Changes in user name
- Maintenance of user name list and passwords
- Program initiation
- Program termination

The following events will **not** be recorded:

- Changes to template files
- Creation of new template files
- Sorting
- Selecting

- Creating a database
- Adjustments to printing parameters
- Saving and restoring of printer layout files
- Changes to the printer configuration
- Accesses to the Help System

## Record Formats

Records are written as ASCII strings terminated with a carriage return/line feed character pair. The fields in the record are separated by commas. String fields containing either a double quote or a comma will be delimited by double quotes. The embedded double quote will be duplicated. Numbers and strings not containing a double quote or comma will not be delimited. To clarify this, an example record is shown:

1,Monday,"December 25th, 1995","It's ""Xmas"" day",12345

Generally, a record contains a record type number, a textual type name, a numeric time flag, the computer time and date of the event, the current user and the related data, if any. The record type numbers are detailed below. The time and date field is in the format **mm/dd/yy** and **hh.mm.ss** respectively and the items are not separated by a comma.

When a log record involves an operation on a database or company data record, only the fields which changed in the record are included in the log. Unchanged fields will be depicted by consecutive commas. This means that there will always be the same number of comma separated fields, but some may be empty.

Each record always consists of 20 fields (an unlimited number if the Dossier Feature is enabled), even though some fields may be empty. **To maintain compatibility with earlier versions of the program, if the Dossier Feature is enabled, at least 50 fields will always be output, even if less than that are used. If more than that are used, then the exact number are output.** The record types are split into groups to facilitate easy report generation. The type of groups is as follows:

### *Group 1 Program Operations (Types 1 - 10)*

Type 1 - Start of Program

Type 2 - End of Program

Type 3 - Logging Start

Type 4 - Logging Stop

Type 5 - Open a Database

Type 6 - Close a Database

### *Group 2 Template Operations (Types 11 - 20)*

Type 11 - Revise a Template Layout

Type 12 - Add a Template

Type 13 - Change a Template

Type 14 - Reassign Template Fields

### *Group 3 User Name Operations (Types 21 - 30)*

Type 21 - Add a User Name

Type 22 - Delete a User Name

Type 23 - Change a Password

Type 24 - Change a User Level

Type 25 - Log in

### *Group 4 Company Information (Type 31 - 40)*

Type 31 - Change Company Data

### *Group 5 Badge Database Operation (Type 41 - 50)*

Type 41 - Add a Badge

Type 42 - Delete a Badge

Type 43 - Change a Badge

Type 44 - Print a Badge

Type 45 - Print Selected Badges

Type 46 - Change Database Fields

Type 47 - Cancel Printing

Type 48 - Print Complete

Type 49 - Signature Capture

### Group Formats

Each record consists of a standard set of initial fields as follows:

*<Type Number>,<Type String>,<Time stamp>,<Date>,<Time>,<UserName>,.....*

| | |
|---|---|
| *<Type Number>* | the type number from the above list |
| *<Type String>* | the textual name of the type (this is locale sensitive) |
| *<Time stamp>* | A numeric value of the number of seconds from 1/1/1970 |
| *<Date>* | The date in the form **mm/dd/yy** |
| *< Time>* | The time in the form **hh:mm:ss** |
| *<UserName>* | The user name, if any |

Where this field group is used below, it is signified by *<header>*

### *Group 1 Program Operations*

This group of records has the format:

*<header>,[<database name>]*

Where *<database name>* is in the fully formed file name of the .IDC file.

This group has the following field usage:

1? Program Initiation (record type 1)
*<header>*
eg: 1,Program Start,799083139,4/28/95 11:32:19,,,,,,,,,,,,,,

2? Program Termination (record type 2)
*<header>*
eg: 2,Program End,799106310,4/28/95 17:58:30,MASTER,,,,,,,,,,,,

3? Logging started (record type 3)
*<header>*
eg: 3,Log Start,799083203,4/28/95 11:33:23,,,,,,,,,,,,,,

4? Logging stopped (record type 4)
*<header>*
eg: 4,Log End,799083168,4/28/95 11:32:48,,,,,,,,,,,,,,

5? Open a Database (record type 5)
*<header>*<database name>
eg: 5,Open Database,799083382,4/28/95 11:36:22,MASTER,E:\ID_CARDS\SAMPLE\SAMPLE.IDC,,,,,,,,,,,,

6? Close a Database (record type 6)
*<header>*<database name>
eg: 6,Close Database,799083386,4/28/95 11:36:26,MASTER,E:\ID_CARDS\SAMPLE\SAMPLE.IDC,,,,,,,,,,,,

### *Group 2 Template Operations*

This group of records has the format:

*<header>,<template_ID>*

Where *<template_ID>*is in the form of a number,space,dash,space and a name.

This group has the following field usage:

1? Revise a template layout (record type 11)
*<header>*<template_ID>
eg: 11,Revise Template,799083419,4/28/95 11:36:59,MASTER,0 - sample,,,,,,,,,,,,

2? Add a Template to the List (record type 12)
*<header>*<template_ID>
eg: 12,Add Template,799083438,4/28/95 11:37:18,MASTER,5 - test7,,,,,,,,,,,,

3? Change a Template in the List (record type 13)
*<header>*<template_ID>
eg: 13,Change Template,799083461,4/28/95 11:37:41,MASTER,4 - test6,,,,,,,,,,,,

4? Reassign Fields on a Template (record type 14)
*<header>*<template_ID>
eg: 14,Reassign Fields,799083483,4/28/95 11:38:03,MASTER,0 - sample,,,,,,,,,,,,

### *Group 3 User Name Operations*

This group of records has the format:

*<header>,<name1>,[<name2>],[<user_level>]*

Note that the password is never included.

This group has the following field usage:

1? Add a New User (record type 21)
*<header>*,<user_name>,<user_level>
eg: 21,Add User,799347971,5/1/95 13:06:11,MASTER,COLIN,BROWSE,,,,,,,,,,,

2? Delete a User Name (record type 22)
*<header>*,<user_name>
eg: 22,Delete User,799348004,5/1/95 13:06:44,MASTER,COLIN,,,,,,,,,,,,

3? Change a Password (record type 23)
<*header*>,<user_name>
eg: 23,Change Password,799347971,5/1/95 13:06:11,MASTER,COLIN,,,,,,,,,,,,

4? Change User Level (record type 24)
<*header*>,<user_name>,<old_user_level>,<new_user_level>
eg: 24,Change User Level,799347990,5/1/95 13:06:30,MASTER,COLIN,BROWSE,SUPERVISORY,,,,,,,,,,

5? Log In (record type 25)
<*header*>,<name>,<user_level>
eg: 25,Log In,799347946,5/1/95 13:05:46,MASTER,MASTER,MASTER,,,,,,,,,,,,

### *Group 4 Company Information*

This group of records has the format:

*<header>,[<company name>],[<street>],[<city>],[<state>],[<zip>],[<country>]*

This group has the following field usage:

1? Change User Data (record type 31)
<*header*>,<company name>,<street>,<city>,<state>,<zip>,<country>
eg: 31,Change User Data,799348677,5/1/95 13:17:57,MASTER,XYZ Corp.,,,,,,,,,,,,
Note: Only fields which changed are included.

### *Group 5 Badge Database Operation*

This group of records has the format:

*<header>,<datakey>,[<photofile>],[<changed>],[<template>],[<field1>],.....[<fieldn>],.....*

where:

| | |
|---|---|
| <*datakey*> | is a number and is always present |
| <*photofile*> | is a file name (eg: EDFRETQW.JPG) |
| <*changed*> | is either an X or nothing |
| <*template*> | is a number and name |
| <*field1*>..<*fieldn*> | are as many string and number fields (up to 10 [at least 50 if the Dossier option is enabled]) as exist and are know to BadgeBuilder® |

When such a record is included below, it will be signified by <DBFIELDS>. Note that only fields which have actually changed are included in the record, unless noted otherwise. Also note that the Print Selected Badges record, has the selection criteria instead of the field data.

This group has the following field usage:

1? Add a Badge Record (record type 41)
<*header*>,<*DBFIELDS*>
eg: 41,Add Badge,799348890,5/1/95 13:21:30,MASTER,799348888,,X,0 - sample,1234,Electrical,Fred Smith,43243,,,,,
Note: All fields will contain data in DBFIELDS as all are considered to have changed

2? Delete a Badge Record (record type 42)
<*header*>,<*DBFIELDS*>
eg: 42,Delete Badge,799348900,5/1/95 13:21:40,MASTER,799348888,,X,0 - sample,1234,Electrical,Fred Smith,43243,,,,,

3? Change a Badge Record (record type 43)
*<header>,<DBFIELDS>*
eg 43,Change Badge,799348896,5/1/95 13:21:36,MASTER,799348888,,,,,,Fred J. Smith,,,,,,,

4? Print a Single Badge (record type 44)
*<header>,<DBFIELDS>*
eg 44,Print Badge,799348930,5/1/95 13:22:10,MASTER,775850350,YLPTSJWE.PCX,,
0 - sample,0002601,Technical,"Jeffries, William",000260,,,,,,

5? Print a Selection of Badges (record type 45)
*<header>,<SQL selection criteria>*
eg 45,Print Selected Badges,799348971,5/1/95 13:22:51,MASTER,"UCASE(""EmployeeName"") LIKE '%LL%'",,,,,,,,,,,,
NOTE: If no criteria exists all badges were printed

6? Change Database Fields (record type 46)
*<header>,<newfieldname_0>,.....<newfieldname_n>*
eg: 46,Change Fields,799349001,5/1/95 13:23:21,MASTER,BadgeNumber,Department,
EmployeeName,EmployeeNumber,NewField,,,,,,,,
Note: All field names in the database will be included.

7? Printing Canceled (record type 47)
*<header>*
eg: 47,Print Canceled,799348972,5/1/95 13:22:52,MASTER,,,,,,,,,,,,,

8? Printing Completed (record type 48)
*<header>*
eg: 48,Print Completed,799348932,5/1/95 13:22:12,MASTER,,,,,,,,,,,,,

9? Signature Captured (record type 49)
*<header>*
eg: 49,Signature Captured,799348932,5/1/95 13:22:12,MASTER,,,,,,,,,,,,,

10? Note: The signature file has the same name as the photograph but with a .SIG extension. It is in the same directory.

# Signature Files

## *Signature File Format*

A signature file content is described by the following pseudo C structure.

```
struct
    {
    long strLength;                 // Number of characters in file title string
    char title[strLength];          // Title string (does not include a terminating NULL)
    long ver;                       // file version number
    long enclosingBox[2][2];        // top left and bottom right coordinates of signature extents
    long pointCount;                // Number of coordinate points in signature
    long points[2][pointCount];     // Array of point pairs containing signature coordinates
    }
```

The signature pen starts down. Any point pair of value (0,0) represents a pen up. A pen up is implied at the end of the data. All points are connected by a straight line.

# Verifier Logs

## *Verifier Logger Records*

When the Verifier feature is used a file called **VERIFIER.LOG** is generated for each verification action. The log file will be placed in the same directory as the main **BB.EXE** program and will be created the first time a verification action is executed, if it does not already exist. It is **never** purged or deleted by BadgeBuilder®. This is the responsibility of the user making use of the log file.

Each record consists of a fixed set of fields as follows:

*<TimeStamp>,<Date>,<Time>,<UserName>,<DatabaseName>,<SearchField>,<SearchString>, <Matches>,<DbField1>,……<DbField10>*

| | |
|---|---|
| *<TimeStamp>* | A numeric value of the number of seconds from 1/1/1970 |
| *<Date>* | The date in the form **mm/dd/yy** |
| *< Time>* | The time in the form **hh:mm:ss** |
| *<UserName>* | The user name, if any |
| *<DatabaseName>* | The name of the IDC file |
| *<SearchField>* | The database field which was searched |
| *<SearchString>* | The character string searched for |
| *<Matches>* | The number of matching records |
| *<DbField1>,……<DbField10>* | The first 10 fields of the matching record, all separating commas will be present, even if the database has less than 10 fields |

---

# Revision Information

| Specification Revision | BadgeBuilder® Version |
|---|---|
| 1.4 | 1.910 |
| 2.0 | 2.0 |
| 2.1 | 2.3 |
| 2.2 | 2.6 |
| 2.3 | 2.61.03 |
| 2.4 | 2.61.03 |
| 2.4 | 2.61.04 |
| 2.5 | 2.61.05 |
| 2.6 | 2.62.00 |
| 2.7 | 3.00.0 |
| 2.8 | 3.3.2 |
| 2.9 | 3.3.3 |
| 3.0 | 3.4.0.0 |
| 4.0 | 4.0 |