REPLY TOOLKIT

FOR REPLY®, REPLY® WW,
REPLY® EZ & REPLY® EU

User Reference Guide

For OCX Version 6.0.15

Updated May 1, 2008

Fleetwood Reply® Standard/Worldwide (WW)/Export (EU)/EZ System
Application Programming Interface (API)
Software License Agreement

IMPORTANT!

PLEASE READ THIS LICENSE AGREEMENT CAREFULLY BEFORE USING THE REPLY WORLDWIDE/STANDARD/EXPORT(EU)/EZ SYSTEM APPLICATION PROGRAMMING INTERFACE ("SOFTWARE").

FLEETWOOD GROUP, INC. ("FLEETWOOD") IS WILLING TO LICENSE THIS SOFTWARE TO YOU ONLY ON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT.

BY ACCEPTING THIS SOFTWARE AND/OR ITS ASSOCIATED FILES AND DOCUMENTATION, YOU AS THE INDIVIDUAL, THE COMPANY, OR THE LEGAL ENTITY THAT WILL BE UTILIZING THE SOFTWARE (REFERENCED BELOW AS "YOU OR YOUR") BECOME BOUND BY THE TERMS OF THIS LICENSE.

THIS IS A LEGAL AND ENFORCEABLE CONTRACT BETWEEN YOU AND FLEETWOOD.

IF YOU DO NOT AGREE TO ABIDE BY ALL OF THE TERMS AND CONDITIONS OF THIS LICENSE, THEN YOU SHALL MAKE NO FURTHER USE OF THIS SOFTWARE, AND YOU SHALL IMMEDIATELY RETURN THE SOFTWARE (PLUS ALL ASSOCIATED FILES AND DOCUMENTATION) TO FLEETWOOD INTACT AND UNUSED, AND YOU SHALL RETAIN NO COPIES OF THIS SOFTWARE OR ITS DOCUMENTATION FOR ANY PURPOSE WHATSOEVER.

THIS SOFTWARE IS PROTECTED BY COPYRIGHT LAWS AND INTERNATIONAL TREATIES. ANY UNAUTHORIZED POSSESSION, REPRODUCTION, OR DISTRIBUTION OF THE SOFTWARE, ITS COMPONENTS, OR ITS DERIVATIVE APPLICATIONS MAY RESULT IN SERIOUS CIVIL AND/OR CRIMINAL PENALTIES.

1. License. The Software that accompanies this License is the sole property of Fleetwood. However, while Fleetwood continues to own the Software, You will have certain rights specified by this License to use the Software after Your acceptance of this License. This License applies to this Software as well as any prior and future releases, revisions, modifications, or enhancements to the Software that Fleetwood may furnish to You.

Subject to the terms below, You are hereby granted the right to use the Software and its associated materials for application development and distribute selected Software controls/elements ("OCX") with Your applications.

This Software is LICENSED, NOT SOLD, to You by Fleetwood for uses only according to the terms of this License Agreement. Fleetwood maintains exclusive and permanent ownership of the Software and reserves any and all rights not expressly granted to You.

This is NOT free Software, and You acquire NO interest in it or its subsets (i.e., computer code, Reply® system commands, computer communications functions, etc.). You only own the media on which the Software is recorded or fixed, but Fleetwood retains sole ownership of the Software itself.

This License Agreement allows You, as the Licensee, to:

a. Use the Software on ONE (1) computer or local workstation. To "use" the Software means that the Software is either loaded in the memory (i.e., RAM) of a computer or installed on the permanent memory of a computer (i.e., hard disk, etc.). One registered copy of the Software may either be used by a single person who uses the Software personally on one computer, or installed on a single workstation used non-simultaneously by multiple people, but not both.

b. Distribute ONE copy of the OCX with each application You develop with this Software ROYALTY FREE.

c. Make ONE copy of the Software in machine readable form solely for backup purposes. As an express condition of this License, You must reproduce on each copy any copyright notice or other proprietary notice that is on the original copy supplied by Fleetwood.

d. Notwithstanding any other terms in this License Agreement, if the Software is licensed as an upgrade or update, then You may only use the Software to replace previously validly licensed versions of the same Software. You agree that the upgrade or update does not constitute the granting of a second license to Software (i.e., You may not use the upgrade or update in addition to the Software it is replacing, nor may You transfer the Software which is being replaced to another party).

2. Restrictions. This Software contains valuable trade secrets and, to protect them:

YOU MAY NOT MAKE ANY ATTEMPT TO DISCOVER THE SOURCE CODE OF THE SOFTWARE. YOU MAY NOT REVERSE ENGINEER, DECOMPILE, DISASSEMBLE OR OTHERWISE REDUCE THE SOFTWARE TO ANY HUMAN PERCEIVABLE FORM. YOU MAY NOT MODIFY, ADAPT, OR TRANSLATE THE SOFTWARE. YOU MAY NOT SUBLICENSE, RENT, SELL, LEASE, OR LOAN ALL OR ANY PORTION OF THE SOFTWARE. YOU MAY NOT CREATE DERIVATIVE WORKS BASED UPON THE SOFTWARE OR ANY PART THEREOF. BASED ON KNOWLEDGE OR USAGE OF THE SOFTWARE, YOU MAY NOT CREATE OTHER REPLY STANDARD/WORLDWIDE (WW) /EXPORT(EU)/EZ SYSTEM CONTROL SOFTWARE THAT REPRESENTS COMPUTER OR MICROCONTROLLER CODE THAT CAN BE OR IS SOLD AS "PROGRAMMING TOOLS" OR “SYSTEM INTERFACES”.

While this license specifies certain rights to distribute components of the Software with applications You may develop from time to time for use with the Reply Standard/Worldwide (WW)/Export (EU)/EZ system, YOU MAY NOT USE OR CONVEY THIS SOFTWARE IN ITS FULL, PARTIAL, DISASSEMBLED, OR CONCEPTUAL FORMS FOR ANY OTHER PURPOSE. ALSO, YOU MAY NOT CREATE, OFFER, OR DISTRIBUTE SOFTWARE AND/OR SYSTEM CONTROLS FOR THE REPLY STANDARD/WORLDWIDE (WW)/EXPORT (EU)/EZ SYSTEM SIMILAR IN FUNCTIONALITY TO THE SOFTWARE CONVEYED BY THIS LICENSE.

3. Dual Media. Even if the Software product includes the Software on more than one medium (e.g., on a CD, on magnetic disks, or as a file sent by email or downloaded from the Internet), You are only licensed to use ONE copy of the Software as described in Section 1. You MAY NOT use the Software stored on the

other medium on another computer or common storage device, NOR may You rent, lease, sell, loan or transfer the Software or its source information or its documentation to another user.

4. Export Law Assurances. Export of this Software is governed by the laws and regulations of the United States and import laws and regulations of certain other countries. You agree that neither the Software nor any direct product thereof is being or will be shipped, transferred or re-exported, directly or indirectly, into any country prohibited by the United States Export Administration Act and the regulations thereunder or will be used for any purpose prohibited by the Act. Export or re-export of Software to any entity on the Denied Parties List and other lists promulgated by various agencies of the United States Federal Government is strictly prohibited. (Note: Other Federal trade rules and regulations may apply to this product and its derivatives from time to time, and You agree to be exclusively responsible for compliance as well as exclusively liable for any noncompliance.)

5. Termination. This License is effective to all users of the Software until terminated by its owner, Fleetwood. This License will terminate immediately without notice from Fleetwood if You fail to comply with any provision of this License Agreement. This License may also terminate upon notice by Fleetwood should You use the Software in a manner Fleetwood, in the exercise of its sole discretion as owner of the Software, deems inappropriate. This License may also terminate immediately by judicial resolution. You may also be subject to damages awarded by court or arbitration proceedings. Upon such termination You must destroy the Software and all copies thereof. Sections 6 through and including 15 of this License will survive any termination of this License.

6. Limited Warranty. The Software is licensed 'as is'. Fleetwood will offer to replace defective Software reported within 30 days of delivery. Technical support for the Software is not included in the License but may be available according to policies established by Fleetwood from time to time. When available, this support is limited to providing Software documentation and explaining component functionality. Support will NOT include advice for or assistance in designing an application that incorporates the Software. The availability and cost of support will vary according to policies that are subject to change without notice.

THIS LIMITED WARRANTY IS THE ONLY WARRANTY PROVIDED BY FLEETWOOD AND ITS LICENSERS. FLEETWOOD AND ITS LICENSERS EXPRESSLY DISCLAIM ALL OTHER WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE WITH REGARD TO THE SOFTWARE AND ACCOMPANYING WRITTEN MATERIALS.

BECAUSE SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

7. Limitation of Remedies and Damages. In no event will Fleetwood, its directors, officers, owners, employees or affiliates of any of the foregoing, past or present, be liable to You or Your application users for any consequential, incidental, indirect, special, or punitive damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information and the like), whether foreseeable or unforeseeable, arising out of the use of or inability to use the Software (or its accompanying materials and Reply hardware), regardless of the basis of the claim and even if Fleetwood or a representative of Fleetwood has been advised of the possibility of such damage. Fleetwood's liability for direct damages for any cause whatsoever, and regardless of the form of the action, will be limited to money damages not to exceed the total amount paid to Fleetwood for the Software License.

The disclaimers and limitations set forth above will apply regardless of whether You accept the Software.

BECAUSE SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

8. Applicable Law. This License Agreement shall be governed for all purposes by the laws of the State of Michigan.

9. Survivorship of Provisions. If any provision of this License Agreement shall be held by a court of competent jurisdiction to be contrary to law, that provision will be enforced to the maximum extent permissible, and the remaining provisions of this License Agreement will remain in full force and effect.

10. Disputes. You agree to resolve any and all disputes or controversies arising from or relating to this License Agreement by arbitration in lieu of legal and/or court action. Arbitration will be conducted in the City of Holland, Michigan, by the American Arbitration Association, by a panel of three or more arbitrators mutually acceptable to the parties, and in accordance with the procedural rules and regulations of the selected association. Fleetwood reserves the right to refuse or cease arbitration and pursue legal and/or court action should the dispute or controversy result from Your violation of this License's terms, particularly if such violation involves (i) unauthorized use or distribution of the Software or other Fleetwood proprietary information or (ii) Your failure to promptly and fully pay any prearranged License fees.

11. Nonassignability. This License and the rights and duties hereunder are personal to the Licensee upon whose skill, judgment, reputation, and form and method of doing business Fleetwood is relying. You MAY NOT transfer this License or any of its rights in whole or in part by assignment, sale, merger, or consolidation, whether by operation of law or otherwise, without the prior written consent of Fleetwood. You MAY NOT rent, lease, sell, loan or transfer the Software or its source information or its documentation to another user under any circumstance except as expressly provided in this license.

12. Trademarks. You ARE NOT authorized to employ the trademarks, trade names, or commercial symbols of Fleetwood without the expressed and advance written approval of Fleetwood, except as specifically authorized by this License Agreement. This License authorizes these two specific uses of such information, PROVIDING You agree that You shall immediately terminate such uses upon notice by Fleetwood Group, Inc.: (1) You MAY use the names "Fleetwood Group, Inc." and "Reply®" in text form to provide necessary reference to the manufacturer and its products in your application's documentation and promotion, providing that ownership of those names is properly credited to Fleetwood Group, Inc., and (2) You MAY hyperlink the web-enabled content of your application documentation and promotion to the www.replysystems.com website, so long as the information contained in Your documentation and promotion is recognized by Fleetwood Group, Inc. to accurately and positively represent itself and its products. You DO NOT acquire any proprietary interest in Fleetwood products, trademarks, trade names, or commercial symbols as the result of using the Software.

13. Status of Parties. You are acting pursuant to this License Agreement in the capacity of an independent contractor. The management of Your business, including the formulation and execution of plans, policies, and procedures, are Your sole prerogative and responsibility. You SHALL NOT use any Fleetwood trademark, trade name, or commercial symbol as part of the name under which you conduct business, and nothing contained in this License shall be interpreted or construed so as to characterize the relationship between You and Fleetwood as a joint venture, partnership, agency or franchise for any purpose whatsoever.

14. Reservations. Fleetwood reserves the right to appoint other Licensees of this Software within any geographic area, recommend applications, establish and adjust License fees, and modify or discontinue the distribution of the Software or any product it supports, all without incurring any obligation of any kind

whatsoever to You or other Licensees. Additionally, Fleetwood reserves any and all rights not expressly granted to You.

15. Binding Effect. This License Agreement shall be binding upon the parties hereto and upon their respective executor, administrators, legal representatives, successors, and assigns.

<END OF LICENSE>

TABLE OF CONTENTS

USER GUIDE

GENERAL OVERVIEW

The Reply® wireless response system is comprised of wireless handheld units, a base station receiver/transmitter, and a control computer. The control computer connects to the base station via a serial or Ethernet connection. Commands are issued from the control computer to the handsets via the base station. The ActiveX Toolkit will allow software developers to create customized applications for communicating with the Reply® wireless response system. The ActiveX Toolkit encapsulates communication between the Reply® base station and the control computer. This is intended to foster an open architecture concept for the Reply® system while simultaneously simplifying implementation.

TOOLKIT INSTALLATION

The Reply® Toolkit is contained in Reply.ocx. All runtime libraries are compiled into the ActiveX control. In order to use the toolkit in your application, it must be registered. This can be accomplished by using RegSvr32.exe in Windows. For example, the following command registers the Reply® Toolkit under a typical Windows XP installation.

```
c:\winnt\system32\regsvr32.exe "c:\Program Files\Fleetwood Group\Reply\Reply.ocx"
```

The toolkit can be uninstalled by issuing the same command with a "–u" argument. The Reply.ocx file can then be deleted.

To use the Reply® Toolkit in your application, it must be referenced in the project in your development environment. For example, in Visual Basic 6, with an open project, select the Project -> Components… menu item. Select Reply ActiveX Control from the list of controls. The Toolbox (control palette) will now contain the control ReplyX. This control can be used at design time by adding it to a form or container object from the toolbox or it can be instantiated within code under most programming languages.

CONNECTING TO BASE STATION

Note: To make changes to the communication or model settings, the Reply instance must be disconnected. Reply.Connected contains the current connection status (True / False).

To connect to the base station, begin by selecting the correct ReplyModel and SerialPort (default port is 1). The SerialPort property accepts values between 1-16. You can then call Reply.Connect(). If a connection failure occurs (base not found, base already connected, base not powered on, etc.) an error event or exception (see ErrorEventsEnabled property) will be raised that the base is busy or cannot be found.

SPECIFYING KEYPADS

The Reply control maintains a list of keypads that actions are performed against. This allows a developer to control which individual keypads receive data. To specify a keypad or range of keypads, use the AddKeypad method. AddKeypad accepts two parameters, a starting keypad number and a count. Adding a range of keypads from 1 – 100 is as simple as calling AddKeypad(1, 100). Suppose you are a nerd apprentice loyal to your master Fibonacci and want to only communicate with keypads 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 & 233. You would call the following:

```
myBase.AddKeypad(1, 1) // keypad 1
myBase.AddKeypad(1, 3) // keypads 1 – 3
myBase.AddKeypad(5, 1) // keypad 5
...
myBase.AddKeypad(233, 1) // keypad 233
```

Note that inserting keypad 1 twice does not add the keypad to the list twice. The second AddKeypad simply sets the keypad 1 flag again. A subsequent call to RemoveKeypad(1, 1) would remove all reference to keypad 1 and keypad data will not be processed for that keypad.

Other methods of managing keypads exist. AddKeypadByString(sKeypads) accepts a string of the following format "1, 3, 5-10, 20-30, 32". Keypad numbers separated by commas are treated as individual keypads while keypads separated by a dash are interpreted as an inclusive range. Invalid string formats will raise an exception. For example, the string "1-10, 20-" will result in an exception containing an invalid range error.

The keypads can be removed from the KeypadList by calling RemoveKeypad(iKeypad, iCount) or RemoveKeypadByString(sKeypads). To clear the entire KeypadList, call ClearKeypadList().

To determine if a keypad is already in the list, call IsKeypadInList(Keypad #).

HANDLING DATA EVENTS

To handle data events, an event handler must be assigned to the various Reply events. If an event handler is not assigned to a particular event type, your application will not receive that type of event.

For a complete list of all Reply event types and specific data event details, refer to the technical reference portion of this document.

RECEIVING KEYPAD DATA

Keypad data values are sent to your application via the OnKeypadDataReceived event.

Suppose you wish to poll keypads 1-100 during a meeting.

Your application behaves as follows:

1) Assign an event handler to OnKeypadDataReceived

2) reply.AddKeypad(1, 100); // add keypads 1 – 100 to keypad list

3) reply.Connect(); // connect to the base

4) reply.StartPolling(); // begin receiving data events

5) Collect data by processing OnKeypadDataReceived events

6) reply.StopPolling() // stop receiving events

7) reply.Disconnect() // Note: this method implicitly calls StopPolling()

8) Display results of polling

REPLY ACTIVEX CONTROL

PROPERTIES

BaseChannel (* mReply2005)

Specifies the communication channel the base is currently utilizing for communication. This property is readable within 2 seconds of connecting to the base station. Setting this property will change the base station communication channel.

BaseSerialNumber (* mReply2005 / read-only)

Returns the serial number of the base station.

* Note: This property is readable within 2 seconds of connecting to the base station.

BaseVersion (read-only)

Returns the firmware version of the base station that the control is connected to.

* Note: Must be connected in order to read this property. BaseVersion is populated during the Connect() method execution. BaseVersion will be set before Connect() completes and before the Connected property reports true.

BaseVersion has three general states:

- "Not Connected"
- "Legacy Version" (older bases are unable to report their firmware version on command)
- Model & Firmware Version (i.e. "CRS930 0.5.1")

CommunicationTimeout

Specifies the number of milliseconds to wait while detecting a communication failure before raising an error / exception. Default value is 3000 which should work fine for most uses.

Connected (read-only)

The status of the Reply connection. Use *Connect()* and *Disconnect()* methods to connect to the base.

Note: If the base becomes disconnected, the OCX will issue an error exception and initiate a Disconnect() call. Once the base has been reconnected, the developer must call Connect() manually to reconnect to the base. This is done to prevent the OCX from entering an unknown connection state.

ControlVersion (read-only)

The ControlVersion property returns the major version, minor version, and build number of the Reply control. This property is useful in determining if you are using the latest release of the control and also helps streamline technical support communication.

DebugEventsEnabled

When set to true, the Reply ActiveX will raise OnDebug events, which return text strings reporting the status of the control at various points while connecting, communicating, and disconnecting. The OnDebug event is primarily intended for use in conjunction with a technical support request. A technical support representative may request that you collect specific debug information to help resolve your case.

ErrorEventsEnabled

When set to true, errors events are raised via the OnError event handler assigned by the developer. Error Events are provided as a convenience to developers who prefer to work with events rather than exceptions. Note: developers using events must perform their own error flow control logic, whereas those using exceptions can typically use the exception handling ability built into most programming languages.

ErrorExceptionsEnabled

When set to false, Error Exceptions are suppressed. This is provided as a convenience to developers who prefer to work with events or handle events manually using LastErrorCode & LastErrorMessage. Note: developers using events must perform their own error flow control logic, whereas those using exceptions can typically use the exception handling ability built into most programming languages.

KeypadAddressChannelShortcutsEnabled (* mReply2005)

When set to true, the keypad shortcuts for changing the keypad address and channel are enabled. The false, these shortcuts are disallowed.

*Note: This property will only affect keypads when *KeypadConfigurationMode* = *kpcBase* and when the base is connected.

KeypadConfigurationMode (* mReply2005)

KeypadConfigurationMode has two possible values: kpcManual & kpcBase. When set to kpcManual, all keypad changes must be made manually using the keypad membrane. When set to kpcBase, all keypads that are within range and powered up will be automatically reconfigured based on the Keypad****** properties (these properties are all prefixed with Keypad).

*Note: This property will only affect keypads when *KeypadConfigurationMode* = *kpcBase* and when the base is connected.

KeypadDigitEcho (* mReply2005)

When set to true, a key that is pressed will display the value of the key on the keypad display. For example, with digit echo on, pressing key 2 displays "2" on the keypad display.

*Note: This property will only affect keypads when *KeypadConfigurationMode* = *kpcBase* and when the base is connected.

KeypadGlobalConfigurationLock (* mReply2005)

When set to true, keypad manual configuration will be locked. No shortcuts will work on the keypad. This setting is intended to prevent keypad users from inadvertently or intentionally changing keypad settings.

*Note: This property will only affect keypads when *KeypadConfigurationMode* = *kpcBase* and when the base is connected.

KeypadList (read-only)

Returns a comma-delimited string of keypads. Use *AddKeypad* and *RemoveKeypad* methods to maintain *KeypadList* property.

KeypadListCount (read-only)

Returns the number of keypads represented in the *KeypadList* property.

KeypadLowBatteryNotification (* mReply2005)

When set to true, an OnLowBatteryWarning event will be raised when a keypad experience low battery power. If this property is false, battery level can still be monitored via the OnKeypadTimestamp and OnKeypadSerialNumber events.

*Note: This property will only affect keypads when *KeypadConfigurationMode* = *kpcBase* and when the base is connected.

KeypadOnlyAcceptInitialResponse (* mReply2005)

When set to true, only the keypress will be sent to the base. This property is easy to misunderstand. To clarify, if this property is enabled a keypad user presses 4 can press clear and change their response before the base reads the keypress value. The user cannot press 4, then press 2 before the base has read the keypress value. The 2 will be rejected by the keypad. Once the 4 has been read, the user can then press 2 and it will be accepted by the keypad.

*Note: This property will only affect keypads when *KeypadConfigurationMode* = *kpcBase* and when the base is connected.

KeypadPostAcknowledgement (* mReply2005)

When set to true, a key that is pressed will display horizontal bars on the keypad display. When the base has acknowledged receipt of the key press, the bars will disappear.

*Note: This property will only affect keypads when *KeypadConfigurationMode* = *kpcBase* and when the base is connected.

KeypadSoftkeysEnabled (* mReply2005)

When true, the softkeys at the top of the keypad membrane will be responsive to user actions. The default base behavior sets these softkeys to inactive.

*Note: This property will only affect keypads when *KeypadConfigurationMode* = *kpcBase* and when the base is connected.

KeypadStarKeyEnabled (* mReply2005)

When set to true, the * keypad key will be enabled. By default the * key is disabled. When enabled, pressing the * key will generate an OnKeypadNotification event.

*Note: This property will only affect keypads when *KeypadConfigurationMode* = *kpcBase* and when the base is connected.

KeypadTransmitPowerLevel (* mReply2005)

The default setting is powHigh. This allows keypads to transmit at full power. Other settings include powLow, powLowMid, and powMidHigh. Normally full power is appropriate and results in the longest range of RF communication. Occassionally when interference is a problem, experimenting with lower power levels will yield less RF interference at the expense of keypad range.

*Note: This property will only affect keypads when *KeypadConfigurationMode* = *kpcBase* and when the base is connected.

LastErrorCode (read-only)

Stores the Error Code of the most recent error encountered. A value of zero represents no error. Can be cleared using *ClearLastError* method. Used in conjunction with *LastErrorMessage* property.

LastErrorMessage (read-only)

Stores a description of the most recent error encountered. Empty string represents no error. Can be cleared using *ClearLastError* method.

MaxKeypads (read-only)

Specifies the maximum number of keypads that the system will handle. Value varies by ReplyModel specified.

ReplyModel

The Reply ActiveX Control is capable of communicating with three models of Reply product: Reply, ReplyEZ, and ReplyEU. Each product supports different keypad ranges and exhibits slightly different underlying behaviors. By selecting the correct model of Reply product, the ActiveX control can seamlessly allow your application to communicate with the device.

mReply2005 is the latest version of Reply base / keypad hardware. It supports properties, methods, and events not offered in previous Reply versions. These new properties are noted with "* mReply2005" within the documentation.

Specifying mAutoDetect for the ReplyModel allows the OCX to detect the Reply model from the version reported by the base. Auto detection works for Reply model's 930 and 940.

Note: The ReplyModel property must be set prior to calling Connect().

Disclaimer: If the incorrect ReplyModel is specified for legacy bases (pre-930), results may be unreliable and data loss could occur. If the incorrect ReplyModel is specified for more recent bases, the Connect() call will fail and prevent an incorrect connection.

SerialPort

Specifies the PC communications port to use for serial communications. This property must be set if *CommType* is set to *ctSerial*. Values range from 1 to 99 (default: 1).

Note: Must be disconnected to set this property. The control will raise an exception if an attempt is made to change this property while connected.

Note: The port value of your Reply device can be located using the Windows device manager.

AvoidWifi (* mReply2005, CRS941 and above)

Sets the system to avoid WIFI channels that may cause RF interference during a voting session.

Note: You must be connected to set this property. The property is stored in the base station and will only be changed on the setting of this property.

The CRS941 and above systems are frequency hopping to avoid interferences from other products. However, a heavily used wireless internet access point can make certain frequencies more difficult to transmit on. There are three commonly used channels for Wi-Fi access points: Channel 1, 6 and 11. The Reply® CRS941 and above systems can be set up to avoid 1 to 2 of these channels if needed. The available settings are displayed in Table 1. See your network administrator for help in determining the optimum setting.

Table 1. Wifi Channel Avoidance Settings

Avoid Wi-Fi Setting	Channels can be avoided
None	Use all available frequencies
Low	Avoid 2425 to 2475MHz (avoid Wi-Fi channels 6 and 11)
LowMid	Avoid 2450 to 2475MHz (avoid Wi-Fi channel 11)
LowHigh	Avoid 2425 to 2450MHz (avoid Wi-Fi channel 6)
Mid	Avoid 2401 to 2425 and 2450 to 2475MHz (avoid Wi-Fi channels 1 and 11)
MidHigh	Avoid 2401 to 2425MHz (avoid Wi-Fi channel 1)
High	Avoid 2401 to 2450MHz (avoid Wi-Fi channels 1 and 6)

AddKeypad (Keypad#, Count)

Call this method to add a keypad to the Keypad list. Calling this method with the *Count* parameter will add *Count* keypads sequentially (e.g. *AddKeypad(5, 3)* will add keypads 5,6,7). Count is optional, with a default value of 1.

AddKeypadByString (sKeypads)

Call this method to add keypads to the Keypad list by passing a formatted string. This method accepts a string of the following format "1, 3, 5-10, 20-30, 32". Keypad numbers separated by commas are treated as individual keypads while keypads separated by a dash are interpreted as an inclusive range. Invalid string formats will raise an exception. For example, the string "1-10, 20-" will result in an exception containing an invalid range error.

BeginQuestion(QuestionNumber) {* mReply2005}

Calling this method initiates timestamp mode and stores the optional QuestionNumber parameter to be returned with OnKeypadTimestamp events. The QuestionNumber parameter is not meaningful to the base or OCX, it is simply included as a convenience for programmers wishing to better track which question a timestamp is associated with. To read timestamp values, call RequestKeypadTimestamps().

* Note: Calling BeginQuestion will clear the Serial Number buffer in the base.

ClearDataBuffer ()

Call this method to clear the base station's data buffer. Note: Must be connected to call *ClearDataBuffer*.

ClearKeypadList ()

Call this method to clear the entire contents of the *KeypadList*. Note: No OnKeypadDataEvents will be raised until keypads are added back to the *KeypadList*.

ClearLastError ()

Call this method to clear most recent error information from *LastErrorCode* & *LastErrorMessage*.

CollectKeypadSerialNumbers() {* mReply2005}

Calling this method initiates collection of serial numbers from all responding keypads.

To read keypad serial numbers, call RequestKeypadSerialNumbers().

* Note: Calling CollectKeypadSerialNumbers will clear the Timestamp buffer in the base.

Connect (): Boolean

Call this method to establish a connection to a base station. The Connect method will block program execution until a connection to a valid Reply® base has been confirmed. If the connection method succeeds, true is returned, otherwise false is returned. If false, error information is available by checking the values of LastErrorCode and LastErrorMessage or from an Error Event or Exception. Current connection state can be queried by checking *Connected* property. If the control is already connected, the method exits and an OnDebug reminder event is raised. ** Previously an error or exception was raised.

Disconnect ()

Call *Disconnect*() to close a connection to a base station Current connection state can be queried by checking *Connected* property. If the control is already disconnected, this method has no effect.

IsKeypadInList(keypad#) : Boolean

Returns a boolean value that represents whether a keypad is in the list held by the *KeypadList* property. If the *KeypadList* argument is given, then the method will use that list instead.

IsKeyLocked(key#: String) : Boolean

Returns a boolean value that represents whether the specified key is locked or unlocked. A return value of True means that the key is inactive on the keypad device. Use *LockKey()* and *UnlockKey()* to change key lock settings. *Key*# must be in the range "0"-"9" or "*". Note: IsKeyLocked() only returns the current key state in the ActiveX control. The actual state of keys on a given keypad are not accessible via this method.

LockKey(key#: string)

Sets the specified key to inactive on the keypad device. Use *IsKeyLocked()* to determine current key lock settings. *Key*# must be in the range "0"-"9" or "*". The Lock / Unlock status is applied globally to all keypads, without regard to the contents of KeypadList property.

LockAllKeys()

Locks all keys on all keypads (all numeric keys and * key – clear key is never locked).

RemoveKeypad (keypad#, count)

Call this method to remove a keypad from the Keypad list. Calling this method with the *count* parameter will remove *count* keypads sequentially (e.g., *RemoveKeypad(5, 3)* will remove keypads 5,6, and 7).

RemoveKeypadByString (sKeypads)

Call this method to remove keypads from the Keypad list. Calling this method with an *sKeypads* value of "2, 5-7, 9" will remove keypads 2, 5, 6, 7, and 9.

RequestKeypadSerialNumbers() {* mReply2005}

Calling this method causes the base to send all serial numbers it has collected from all responding keypads. Serial numbers are returned to the program via the OnKeypadSerialNumber event.

* Note: Call CollectKeypadSerialNumbers() to begin collecting keypad serial numbers.

RequestKeypadTimestamps() {* mReply2005}

Calling this method causes the base to send all timestamps it has collected from responding keypads. Timestamps are returned to the program via the OnKeypadTimestamp event.

* Note: Call BeginQuestion() to reset the timestamp timer and begin collecting keypad timestamps.

StartPolling ( )

Call this method to initiate polling of keypads.

StopPolling ( )

Call this method to end polling of keypads. No additional OnKeypadDataReceived events will be raised and any keypresses made after polling is stopped will be discarded.

UnlockKey(key#: string)

Restores the specified key to active on the keypad device. Use *IsKeyLocked()* to determine current key lock settings. *Key#* must be in the range "0-"9 or "*". The Lock / Unlock status is applied globally to all keypads, without regard to the contents of KeypadList property.

UnlockAllKeys()

Restores all keypad keys to active on the keypad device.

Note: In the interest of performance, disable DebugEvents prior to building production release software.

OnDebug(Message: string)

Event that returns text strings reporting the status of the control at various points while connecting, communicating, and disconnecting. The OnDebug event is primarily intended for use in conjunction with a technical support request. A technical support representative may request that you collect specific debug information to help resolve your case.

If you are concerned that there is an internal malfunction or are merely curious, set the DebugEventsEnabled property to true, handle this event and watch the messages it reports. It will indicate if communication is occurring between an application and the base station.

OnError(ErrorCode: integer, Description: string)

Alternative to error exceptions being raised for run-time errors. If the ErrorEventsEnabled property is true and the ErrorExceptionsEnabled property is false, the ActiveX control will raise OnError events instead of exceptions when communication with the base is lost. Note that some exceptions will still be raised if the ActiveX is given bad data (in this manner, the ActiveX helps the developer discern design errors from run-time errors). **See Troubleshooting section for a list of ErrorCodes and Descriptions.

Keep in mind that OnError behaves differently than using exceptions. Exceptions alter program execution; you must implement your own error flow control if you choose to use the OnError event. For example, you can either write a method using exceptions:

```
void function myFunction()
{
  try {
    reply.Connect();
    reply.AddKeypadsByString("1-5000"); // will raise exception (out of range)
    . . .
  }
  catch {
    // recover from exception
  }
}
```

Or, you can set the ErrorEventsEnabled property to true amd handle the OnError events.

```
boolean bContinueExecution; // instance variable for flow control

void function myFunction()
{
```

```
  bContinueExecution := true;

  reply.Connect();

  if (bContinueExecution) // value could go to false from OnError event
    reply.ResetBase(); // command only executes if bContinueExecution is true
}

void eventHandler OnError (ErrorCode, Description)
{
  if (ErrorCode = -1) bContinueExecution = false;
}
```

OnKeypadDataReceived (KeypadID: integer, Value: integer)

The OnKeypadDataReceived event that returns the keypad ID and an integer value representing the keypress.

Note: Data events are only raised when a keypress is detected. No data events are raised for a keypad when no keypresses have occurred (i.e. there is no default indicator of keypad inactivity).

OnKeypadNotification (KeypadID: integer)

The OnKeypadNotification event is raised for each keypad that receives a keypress on the * key. Reply systems prior to version 2.1 may not support OnKeypadNotification.

OnKeypadSerialNumber(KeypadID: integer, SerialNum: String, Version: String, BatteryLevel: integer, EndOfSerialNumbers: boolean) {* mReply 2005}

Event returns a text string containing the serial number of the represented by KeypadID. The version parameter represents the version of the code within the keypad unit. BatteryLevel signifies how much battery life remains on a scale of 1 – 4. The EndOfSerialNumbers flag signifies when the last keypad serial number event has been received.

OnKeypadSerialNumber events are received in a block consisting of one event for each keypad that has reported it's serial number. OnKeypadSerialNumber events are initiated by calling RequestKeypadSerialNumbers().

* Note: If only one event is received with a KeypadID = 0 and EndOfSerialNumbers = true, then no keypads reported their serial numbers. Keypads may be off or on a different channel from the base.

OnKeypadTimestamp(KeypadID: integer, QuestionNumber: integer, Timestamp: integer, Version: String, BatteryLevel: integer, EndOfSerialNumbers: boolean) {* mReply 2005}

Event returns a QuestionNumber integer and a Timestamp representing the number of milliseconds elapsed between the call to BeginQuestion and the keypress made by the keypad user on the keypad membrane. *Note: Timestamp is based on 50-millisecond blocks and will always be a factor of 50.

The version parameter represents the version of the code within the keypad unit. BatteryLevel signifies how much battery life remains on a scale of 1 – 4. The EndOfTimestamps flag signifies when the last keypad timestamp event has been received.

OnKeypadTimestamp events are received in a block consisting of one event for each keypad that has reported a timestamp. OnKeypadTimestamp events are initiated by calling RequestKeypadTimestamps().

* Note: If only one event is received with a KeypadID = 0 and EndOfTimestamps = true, then no keypads reported a timestamp. Keypads may be off or on a different channel from the base or RequestKeypadTimestamps() was called before any keypad users could press a key.

OnLowBatteryWarning (KeypadID: integer)

This event returns keypad id of a keypad reporting low battery.

OnSoftkeyDataReceived(KeypadID: integer, KeyValue: String) {* mReply 2005}

Event returns a text string containing "<", "=", or ">" signifying the softkey pressed on the keypad represented by KeypadID.

Reply2005.OCX version 6.0.1 and later have been tested within the .NET programming environment.

To incorporate the OCX within your .NET application, refer to the following steps:

1) Create a new application project within your .NET solution

2) Select Project > Add Reference…

3) Select the COM tab

4) Locate Reply2005 ActiveX Control. Highlight it and click the Select button. Then click OK.

   * Note: If the Reply2005 control is not listed, you will need to install it again using regsvr32.exe.

5) To place the Reply2005 control within your toolbox, right-click on the current toolbox tab (or add a new tab) and click Customize Toolbox.

6) Locate ReplyX control within the Reply2005 library and check the box. The control icon will now appear within on toolbox tab.

Error Codes

Refer to OnDebug event description for additional troubleshooting ideas.

-1	Connection Error (wrong port, base disconnected / off, etc.)
-2	Process Error (must either connect or disconnect before issuing command)
-3	-- currently unused --
-4	Invalid value / range
-5	Invalid data from base (reset base – may signify hardware problem)

When contacting technical support, please include all of the following information:

- Reply version (see ControlVersion property)
- Base station version (see BaseVersion property)
- Operating system (the operating system you are using)
- Development language and version (for example, Visual Basic 6 - service pack 2)
- Exact error message(s)
- Description of steps leading to error (what properties / methods led to error)

REVISION HISTORY

REPLY OCX REVISION

Version 3.0.1

(Initial Production Release)

Version 5.0.2

(Renumbered production release)

Version 5.0.3

- Fixed locking of * key in conjunction with LockAllKeys and UnlockAllKeys
- Improved performance of individual key unlocking – removed redundant unlocks
- Modified LockKey, UnlockKey, and IsKeyLocked parameter values

Version 5.2.1

- Added properties – LastErrorCode & LastErrorMessage
- Added method – ClearLastError()
- Added exception property – ErrorExceptionsEnabled
  Note: This new property means that Exceptions and Error Events are no longer mutually exclusive.
- Modified KeypadList property (read-only, returns comma-delimited list of keypads)
- Fixed recursive exception on failed connect > successful connect > disconnect
- Modified Connect() method to block until connection is verified – boolean result reflects success / failure.
- Modified Connect() so duplicate calls on already connected instance no longer raises an Error. Instead an OnDebug reminder event is raised and the method exits returning false.
- Updated BaseVersion property to more accurately report legacy bases

Version 6.0.1

Added support for new model of Reply hardware

- Added detailed explaination for ErrorCodes in Troubleshooting section of document
- Added new properties
  - ReplyModel (added option mReply2005)
  - BaseChannel
  - BaseSerialNumber
  - KeypadDigitEcho
  - KeypadStarKeyEnabled
  - KeypadLowBatteryNotification
  - KeypadOnlyAcceptInitialResponse
  - KeypadConfigurationMode
  - KeypadAddressChannelShortcutsEnabled
  - KeypadGlobalConfigurationLock

    - KeypadSoftKeysEnabled
    - KeypadTransmitPowerLevel

- Added new methods
    - BeginQuestion()
    - CollectKeypadSerialNumbers()
    - RequestKeypadSerialNumbers()
    - RequestKeypadTimestamps()

- Added new events
    - OnSoftkeyDataReceived
    - OnKeypadTimestamp
    - OnKeypadSerialNumber

Version 6.0.2

- Fixed bug with BaseVersion property

Version 6.0.7

- Expanded support for COM ports 1 – 99
- Added Reply model AutoDetect capability
- Added detection of lost base connection during non-polling periods
- Added blocking for read of BaseVersion property
  (version is read before Connect method returns and Connected property reads True)

Version 6.0.8

- Fixed filtering on KeypadList for keypad notification and softkey events

Version 6.0.9

- Fixed bug reading large numbers of timestamps and serial numbers

Version 6.0.10

- Improved detection of non-Reply serial devices

Version 6.0.11

- Fixed error handling routines

Version 6.0.12

- Increased message speed

REPLY OCX REVISION (CONT.)

Version 6.0.13

- Fixed error handling routines

Version 6.0.14

- Multiple model matrix

Version 6.0.15

- Added Wifi Avoidance property for the Reply Worldwide USB Base Stick

DOCUMENTATION REVISIONS

In addition to updating documentation to reflect OCX changes listed above, the following documentation changes have been made:

Version 3.0.1 (Initial Production Release)

Version 5.0.3

- Modified LockKey, UnlockKey, and IsKeyLocked parameter values

Version 5.2.1

- Modified descriptions for KeypadList and KeypadListCount to reflect functional changes
- Added detailed explaination for ErrorCodes in Troubleshooting section of document
- Added descriptions for new properties / method
  - LastErrorMessage, LastErrorCode, ClearLastError()
  - ErrorExceptionsEnabled

Version 6.0.1

- Added descriptions for new properties / methods

- Added sample code for using Reply OCX with .NET programming environment

Version 6.0.11

- Changed version and name of this document to reflect that of the OCX itself.

Version 6.0.15

- Added Wifi Avoidance property description
- Changed version and name of this document to reflect that of the OCX itself.