Button Script and Command Reference

# INTRODUCTION

This document will provide the basic reference for writing button scripts, however some things you may need to understand will be in other documents:

* Button scripts can use macros – these macros are outlined in the iCell Macro Guide
* Some commands (e.g. adding a product) require specifying a price formula – these macros are outlined in the iCell Behaviours Guide (ADJUSTMENTS section)
* Button scripts are a simple text file and are deployed via icmsControl and generally imported via a patch file.

The document has four sections

* Description of scripting language
* Commonly used commands and their usage
* A few examples
* A comprehensive list of commands that can be used.

# SCRIPTING LANGUAGE

The scripting language is NOT a programming language, it doesn’t check if your code is correct until it runs it. Macros are not variables, they are just text substitutions. A line of the script may look fine while the macro is unexpanded, but once expanded the macro may cause the line to become invalid. Enabling full command processor logging can help debug these issues as it will show before/after and processing order on each line.

For the most part the script will consist of icell Commands and some basic flow control keywords to allow repeating of sections or testing of values. The icell commands are documented in their own section – but for the purposes of this section I’ll be using the generic ‘Command(Parameters)’ to represent any command.

## BUTTON SCRIPT

The button file consists of a text file containing multiple button scripts each of the following format:

**TEMPLATE** [PRIVATE] Name(Parameter1,Parameter2,Parameter3…)  
**OPTION** [Security,SecurityList,Caption,Style,Sticky,ImageName,Caption]=Optionvalue  
**VAR** [REQUIRED] Parameter=Comma Separated List of Possible Values  
*.. optionally multiple VAR lines***DEFAULT** Parameter=Default Value  
*.. optionally multiple DEFAULT LINES  
.. multiple script lines (comments, commands and flow control) see following sections.***END**

The **TEMPLATE** directive specifies the name of the button script, and delineates the start of a new button. If the button script is marked as PRIVATE (optional) then the script will not be visible in the button options page when selecting a script. The name is simply the name of the script. Notice that because scripts can call other scripts – it is advisable to name any such callable script SUB\_Name so it is obvious it is such a script. The parameters are just simple names that will be used with in the script like macros to pass parameters into the script itself.

For example:

TEMPLATE PRIVATE SUB\_Fast\_TabCard(MultiPart,AskName,Prompt,AddLocation)

The **OPTION** directive allows you to specify several options for the button:

* Security: Specify one cashier security level required to access this button
* SecurityList: specify a comma separated list of cashier security levels to access this button (one must match)
* Caption: Specify the caption for a button using this script
* Style: Specify the style (icmsStyles) for a button using this script
* Sticky: Specify that a button using this script be sticky (i.e. stay down until released by another button)
* ImageName: Specify the image name of a button that uses this script
* Caption: Specify the caption for a button using this script

For example:

OPTION SecurityList=Cashier,Supervisor,Manager,SysAdmin

The **VAR** option can be used optionally multiple times (once for each parameter) to specify if it is a required parameter and to specify the possible values of the parameter.

For example:

VAR REQUIRED EntryMode=Preset,Manual,CardSwipe,Search

The **DEFAULT** option can be used optionally multiple times (once for each parameter) to specify the default value for that parameter.

For example:

DEFAULT EntryMode=Preset

The **END** command ends the script definition allowing for the next one to be started, or the end of file. It has no parameters and is used as follows:

END

## COMMANDS

To execute a normal command the syntax is:

Command(Parameters)

Multiple parameters are separated by a comma:

Command(Parameter1,Parameter2,Parameter3)

If you want to execute a command but skip one of the parameters, leave it blank but include the correct number of commas.

Command(Parameter1,,Parameter3)

If a parameter might have an problem character in it – then the vast majority of commands (some commands have special requirements) you can do the following:

Command(“Parameter,One”,”Parameter,Two”)

Some commands return a value, which can be put in a local macro, to execute one of these and place the result in a macro called {MyMacro} do the following:

MyMacro=Command(Parameters)

Some commands can be sent to a specific named object (found in the object properties in the layout designer) – as follows:

Command(Parameters) -> ObjectName

A command that can be sent to an object, and also use the ‘nickname’ of that where the actual object names are defined in the general/objects section of settings.xml (e.g. receipt, entry) to use these nicknames use the following syntax:

Command(Parameters) -> ^NickName

If a command has no parameters, it is advisable to use the following syntax:

Command()

## COMMENTS

If you wish to have a full line comment to explain what the button script is doing you can do it as follows:

# The following does some amazing stuff

You can create multi line comments:

# The following changes have been made:  
# Added Amazing Function (bug 110) 26/07/2021  
# Fixed an issue (bug 192) 30/10/2020

## FLOW CONTROL

Flow control is a technical term that just means – do stuff multiple times, or do stuff optionally, or go somewhere else to do it.

**FOR:** If you want to repeat the same bit of script multiple times – then you can use the **FOR** keyword as follows (it has companion keywords to help it out which will be shown in bold). Use **FOR** when you want to repeat something a specific number of times.

**FOR** Macro **IS** Start **TO** End  
 *.. do script commands including other flow keywords etc***END FOR**

The Macro here is the name, not the expansion {Macro} … you will most likely use the expansion within the loop to refer to the value of {Macro}.

For example:

For Attempt is 1 to {MaxAttempts}  
 *… attempt something*End For

**WHILE:** If you want to repeat the same bit of script multiple times, but you don’t know how many times you can use the **WHILE** keyword as follows (it also has companion keywords). Use **WHILE** when you want to stop when some condition is met.

**WHILE** [**NOT**] {Macro} Condition TestValue  
 *… repeat something*  
**END WHILE**

The {Macro} used here is a previously defined macro and you are testing the value (unlike the FOR keyword). The optional NOT keyword negates the test.

Condition is one of (=, <, > , <=, >=, <> for equals, less, greater, less or equal, greater or equal or not equal), you can OR or NOT to create a compound condition if you wish.

While "{YN}" = "N"  
 *… do stuff*  
 *… ask a question that returns Y or N*  
End While

Instead of a normal condition you can use the IN/OUT keywords where you test the Macro to be in the list provided or NOT (OUT) in the list.

**WHILE** [**NOT**] {Macro} [**IN**|**OUT**] List,Of,Values  
 *… repeat something*  
**END WHILE**

**IF:** If is almost exactly like **WHILE** except the script inside is executed ONLY ONCE if true, and not at all if false, rather than multiple times.

Just like **WHILE** you can use the normal conditions or the IN/OUT syntax as follows:

**IF** [**NOT**] {Macro} Condition TestValue  
 *… optionally do something***END IF**

**IF** [**NOT**] {Macro} [**IN**|**OUT**] List,Of,Values  
 *… optionally do something*  
**END IF**

For example:

IF {Payment} in Tyro,Spice,PCEFTPOS  
 *.. do stuff*END IF

**CASE**: When you want to execute one of several different script parts depending on the value of a macro – use the **CASE** keyword. Like earlier keywords it has some companion keywords which will be highlighted:

**CASE** {Macro}  
 **WHEN** Value1  
 *… do stuff* **WHEN** Value2  
 *… do stuff* **WHEN** Value3  
 *… do stuff* .  
 .  
 [**OTHERWISE]** *… optionally do stuff when none of the values matched***END CASE**

For example:

Case "Single"  
 When "Single"  
 Drawer(1)  
 When "Multi"  
 Drawer({cashierdrawer})  
End Case

## MISCELLANEOUS script keywords

You can use the **SET** keyword to set a local macro, and possibly do some minor calculations on it. For example:

Set BalancePositive to -1\*{Balance}

You can use the **CALL** keyword to call another script defined elsewhere, for example:

call Maxicash\_Payment(Cougar,{PayAmt})

You can log extra information to the Command Processor Log using the **LOG** keyboard, for example:

Log Kitchen Order dockets to be printed = {KPqty}

If you want to ensure a command is executed even if an error occurs earlier, use the special **ALWAYS** keyword, for example:

Always Clear(MACROS)

# COMMON COMANDS AND USAGE

# SAMPLES

# COMMAND REFERENCE

|  |  |  |
| --- | --- | --- |
| Account | | |
|  | | |
|  | **Usage:** | Name=Account(CREATE|CARD|DELETE|TAG|OWN|SUB,Id|PaymentId|Option,Parameters) or Account(DELETE|TAG,Id,Value) |
|  | |
|  | | |
|  | **Description** | Manage transaction accounts |
|  | | |
|  | **Details:** | for CREATE - creates a normal account manually (no authentication) Name - optional name of the macro to assign the result to (for CREATE and CARD only PaymentId - Payment ID (see [#Authenticate|Authenticate] for details) With 1 extra parameter - creates a Debit account with account Id With 2 extra parameters - creates a generic account with AccountId and Name With 3 extra Parameters - creates a gereric account with AccountId, Name and MemberType With 4 extra parameters - creates a generic account with AccountId,Name,SubAccount,SubName With 5 extra parameters - creates a generic account with AccountId,Name,SubId,SubName,MemberType,optional Ratings Grade for CARD - creates a special credit card account (used for EFTPOS/Credit card) Name - optional name of the macro to assign the result to (for CREATE and CARD only PaymentId - Payment ID (see [#Authenticate|Authenticate] for details) Parameter1 : Entry Type, as specified by the PAN Source ID (e.g. 0:Internet, 1:Telephone, 2:Mail order, 3:Customer, 4:Recurring, 5:Installment) Parameter2 : Card Number - 16 digit credit card number Parameter3 : Expiry Date - 4 or 6 digit (mmyy or mmyyyy) expiry month/year for OWN - takes ownership of produt line items in the current part (function disabled as of bug !!110) Id - account id to give ownership to SELECTED - take ownership of selected item MARKED - take ownershup of marked items ANONYMOUS - take ownershup of unowned items for SUB - assigns a subid and sub name to an existing account, then recreates the account with new key Name - returns the new account id (on success) or the existing one (on failure) Id - account id to assign Parameter1 - new sub id Parameter2 - new sub name DELETE - Deletes account given by Id PAUSE - pauses behaviour processing and tx saving in account commands RESUME - resumes behaviour processing and tx saving in account commands (behaivours will run and tx will be saved ON the resume) TAG - sets or searches account tag for account given by Id to Value Account(TAG,AccountKey,Tag) - tags account with non-unique search tag List=Account(TAG,?,SearchTag) - searches and returns list of accounts with search tag UNIQUE - sets or finds account tag for given unique tag. Account(UNIQUE,AccountKey,Tag) - tags account with UNIQUE find key Value=Account(UNIQUE,?,FindKey) - finds account with given UNIQUE find key |
|  | |
|  | |
|  | | |
|  | |  |
|  | | |
| Act | | |
|  | | |
|  | **Usage:** | Act(TillId,LocationId) |
|  | | |
|  | **Description** | Act as an alternate till and location temporarily |
|  | | |
|  | **Details:** | TillID - of the till to act as LocationId - of the till to act as. |
|  | |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Add | | |
|  | | |
|  | **Usage:** | Var=Add(?|? x Qty1|ProductList|ProductId x Qty1, Qty2, Pricing) |
|  | | |
|  | **Description** | Adjustment to add product to the current receipt/invoice, if credit mode then a return is added |
|  | | |
|  | **Details:** | ProductId - A product ID from [catalog.xml] to add ? - the selected product ProductList - e.g. 1234x9+1111x4+4299x2 Qty1 - Qty of product to add (used when [#Quantity|Quantity] command is used) Qty2 - Qty of product to add, quantities from Productlist, Qty1 and Qty2 are multiplicative Pricing - Relative@Absolute, e.g. -10%@Default/2 |
|  | |
|  | | |
|  | | *Full description of pricing in behaviour documentation If a var is specified, then a price check on the adjustment is performed - without changing anything.* |
|  | | |
| Array | | |
|  | | |
|  | **Usage:** | Var=Array(Param1,Param2,....) |
|  | | |
|  | **Description** | Convert a list of parameters into an array macro |
|  | | |
|  | **Details:** | Paramters, param1 will be accessible as {Var.1} etc. |
|  | | |
|  | |  |
|  | | |
| Attach | | |
|  | | |
|  | **Usage:** | Attach(DrawerId|?) |
|  | | |
|  | **Description** | Attach the logged in cashier to the specified drawer |
|  | | |
|  | **Details:** | DrawerId a number specifying the drawer ? the default configured (via [#Drawer|Drawer] command) drawer. |
|  | |
|  | | |
|  | |  |
|  | | |
| Attribute | | |
|  | | |
|  | **Usage:** | Attribute(Name,Value) |
|  | | |
|  | **Description** | Sets a transaction attribute |
|  | | |
|  | **Details:** | Name - name of the attribute (e.g. NAME for the transaction name) Value - value to set the attribute to. |
|  | |
|  | | |
|  | |  |
|  | | |
| Authenticate | | |
|  | | |
|  | **Usage:** | Name=Authenticate(Payment|Payment AS Alias|@AUTH|@ACCRUE|@XML|@DEFAULT,LookupId) |
|  | | |
|  | **Description** | Authenticates the user with ID, and keeps it in the macro {Name} for later use |
|  | | |
|  | **Details:** | Name - the name of the variable to store the customer in, a special case is Name# - which creates incrementing (Name1,Name2) variables. Payment - a payment interface named and defined in [gateways.xml] Payment AS Alias - as payment, but with an override alias (e.g. EFTPOS as VISA) @AUTH - use the auth interface defined in [gateways.xml] @ACCRUE - use the accrue interface defined in [gateways.xml] @XML - use the xml interface defined in [gateways.xml] @DEFAULT - use the default interface defined in [gateways.xml] LookupId - the customer reference number, card number or ID to attempt to authenticate |
|  | |
|  | | |
|  | | *If the interface specified does not support authenticate, then the local [customer.xml] file is checked* |
|  | | |
| Balance | | |
|  | | |
|  | **Usage:** | Var=Balance(Customer,Bucket) |
|  | | |
|  | **Description** | Return the balance of customer |
|  | | |
|  | **Details:** | Customer - customer as set by [#Authenticate|Authenticate] Bucket - optional bucket id to get balance of. Var - balance as a number, no leading currency symbols. |
|  | |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Banking | | |
|  | | |
|  | **Usage:** | Banking(RECORD,PaymentId,Drawer) or Banking(TILL,PaymentId,Name,Quantity,Per Value) or Banking(CANCEL) or Banking(REPORT,TemplateId,Filename) or Value=Banking(FLOAT,FloatId) or Banking(TAKINGS,FloatId,DrawerId) |
|  | |
|  | | |
|  | **Description** | Perform banking functions at the till, recording counted moneys and then at finalise recorded moneys into a transaction or cancel the current banking tx |
|  | |
|  | | |
|  | **Details:** | for TILL - adds a count value to the banking transaction PaymentId - the payment id to assign this count to, e.g. CASH for coin/note counts Name - A descriptive name for the item, e.g. 50c Pieces Quantity - The count of this item, e.g. 197 (50 c pieces), or 1 for non currency counts such as EFTPOS Per Value - The value per single of this item, e.g. 0.50 (for 50c pieces), or the full amount for non currency for RECORD - finalises and processes the banking transaction PaymentId - the payment Id to include the stored money details from. Drawer - the drawer number to process (0 for manual drawer tills) for CANCEL - no parameters required, clears the current banking transaction for REPORT - saves a report to the report folder Template - Target printer/template Filename - root file name (date/time etc appended) for FLOAT - gets a float from the configured banking vendor Value - Return the net result of the float into a macro FloatId - float id you are requesting for TAKINGS - submits takings from local money record to configured banking vendor FloatId - float id you are returning DrawerId - which drawer you are reporting |
|  | |
|  | | |
|  | |  |
|  | | |
| Base | | |
|  | | |
|  | **Usage:** | Base(?,\*|Qty,Formula) |
|  | | |
|  | **Description** | Set the fixed price (BASE) formula, use this in buttons - rarely in behaviours. |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| Batch | | |
|  | | |
|  | **Usage:** | Batch(PDE,Adjustment,Direction,AdjustmentFile) |
|  | | |
|  | **Description** | Perform a batch adjustment on the active transaction |
|  | | |
|  | **Details:** | Currently PDE is the only available adjustment type Adjustment - e.g. Add or Stock Direction - either +1 or -1 File to get the batch list from (see [#Question|Question] for how to select a file) |
|  | |
|  | | |
|  | |  |
|  | | |
| Cashout | | |
|  | | |
|  | **Usage:** | Cashout(FROM|OF,amount) |
|  | | |
|  | **Description** | Adds a cashout notify to the transaction |
|  | | |
|  | **Details:** | FROM - the cashout notify is from the given amount OF - the cashout notify is of the given amount. |
|  | |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Catalog | | |
|  | | |
|  | **Usage:** | Var=Catalog(ProductId,Macro|Request) |
|  | | |
|  | **Description** | Returns the value of a product entry field |
|  | | |
|  | **Details:** | ProcuctId any viable product ID for a product Macro one of the following: @.NAME - returns the normal name of the product @.BUTTONNAME - returns the button name of the product @.LONGNAME - returns the long name of the product @.PARENT - returns the parent ID of the product @.PRICE.X - returns the dollar price for prcing value X (e.g. PRICE.DEFAULT/1) @.GROUP.X - returns the group for group type X (e.g. GROUP.DEPT) @.FLAG.X - returns YES if flag X exists, NO if it does not. OR Request one of the following: VARIANTS - Returns a list of id=name variants suitable for sending to a pick list |
|  | |
|  | | |
|  | |  |
|  | | |
| Charge | | |
|  | | |
|  | **Usage:** | Charge(TAB|ROOM,\_|Room Id) |
|  | | |
|  | **Description** | TAB : DO a tab charge virtual transaction |
|  | | |
|  | **Details:** | ROOM : Do a room charge to the current room charge system on the server Room Id - the full room id and possibly sub id as required by the active room charge system. |
|  | |
|  | | |
|  | |  |
|  | | |
| Checkpoint | | |
|  | | |
|  | **Usage:** | D=Checkpoint(Name,HOURS|MINUTES|WHEN) or Checkpoint(Name,\_|CLEAR) |
|  | | |
|  | **Description** | Store a checkpoint with the current time and allow the time to be retrieved later |
|  | | |
|  | **Details:** | D is specified retreives the hour duration since the Name is the name of the checkpoint CLEAR - if specified clears the checkpoint HOURS - return hours since checkpoint, returns 'NO' if checkpoint invalid MINUTES - returns minutes since checkpoint, returns 'NO' if checkpoint invalid WHEN - returns formated date/time (e.g. 31-Jan-2017 8:15pm) of checkpoint, returns 'NO' if checkpoint invalid |
|  | |
|  | | |
|  | |  |
|  | | |
| Choose | | |
|  | | |
|  | **Usage:** | Var=Choose(Index,List of Choices) |
|  | | |
|  | **Description** | Choose an item out of a list of choices based upon the index |
|  | | |
|  | **Details:** | Index - an index (0,1,2,3,....) List of choices - comma separated list of choices - index 0 returns the first one, 1 the second etc. |
|  | |
|  | | |
|  | |  |
|  | | |
| Clear | | |
|  | | |
|  | **Usage:** | Clear(MACROS) |
|  | | |
|  | **Description** | Clears MACROS, may be extended to clear other things. |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Console | | |
|  | | |
|  | **Usage:** | Console(Target,OFF|ON|TOGGLE|ALL|CURRENT|SELECTED|LAST,Filter) |
|  | | |
|  | **Description** | As [#Print|Print] except prints to the console (or remote output) |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| Cover | | |
|  | | |
|  | **Usage:** | Cover(Qty,Price) |
|  | | |
|  | **Description** | Add a cover sundry to the receipt, with given price. |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| Credit | | |
|  | | |
|  | **Usage:** | Credit(OFF|ITEM|SALE|ON) |
|  | | |
|  | **Description** | Set the current credit mode status |
|  | | |
|  | **Details:** | OFF - turn off credit mode ON - turn on credit mode (currently not available) SALE - turn on credit mode until the current sale is finalised ITEM - turn on credit mode for one operation (receipt or adjustment) |
|  | |
|  | | |
|  | |  |
|  | | |
| Cut | | |
|  | | |
|  | **Usage:** | Cut |
|  | | |
|  | **Description** | Cut the marked items out of the current transaction into the clipboard |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| CutTo | | |
|  | | |
|  | **Usage:** | CutTo(TableId,CURRENT|NEW) |
|  | | |
|  | **Description** | Cut the marked items out of the current transaction into the specified transaction |
|  | | |
|  | **Details:** | TableId - the target transaction to paste into CURRENT - paste into the current part NEW - paste into a new part. |
|  | |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| DBUpdate | | |
|  | | |
|  | **Usage:** | DBUpdate(PRICING|RESEND,Target,TillList,Name,Value,Expiry) |
|  | | |
|  | **Description** | Send updates to the iConnect database |
|  | | |
|  | **Details:** | PRICING - updates pricing overrides on the iConnect database \* Target : Either product id or \* if reseting pricing \* TillList : \* represents all tills, L<locationid> all tills in a location, C<CellId> all tills with this cellid, <till>+<till>+<till>... a list of tills \* Name : Pricing Level Name \* Value : Either a value (e.g. 4.99) or - for when you are reseting a specific price \* Expiry : Expiry in days for the price override DBUpdate(Pricing,<productid>,TillList,<pricename>,<pricevalue>) - price override DBUpdate(Pricing,<productid>,TillList,<pricename>,-) - reset pricing for specific product and price name DBUpdate(Pricing,<productid>,TillList) - reset pricing for product DBUpdate(Pricing,\*,TillList,<pricename>,-) - reset pricing for all products for given pricename DBUpdate(Pricing,\*,TillList) - reset pricing for all products NAME - updates a name override for a product \* Target : Product id \* TillList : Always \* \* Name : Product Name override, blank to reset the override \* Value : Optional Description override, \\ are converted to CR/LF DBUpdate(Name,1294,\*,Frank Furter) RESEND - initiates a RESEND to tills event in iConnect \* TillList : \* represents all tills, L<locationid> all tills in a location, <till>+<till>+<till>... a list of tills DBUpdate(Resend,Catalog,TillList) - Trigger a catalog resend |
|  | |
|  | | |
|  | |  |
|  | | |
| Delete | | |
|  | | |
|  | **Usage:** | Delete(?,n) or Delete(ALL) to remove all products (reseting the transaction) |
|  | | |
|  | **Description** | Delete n items from the selected item |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| Dettach | | |
|  | | |
|  | **Usage:** | Dettach |
|  | | |
|  | **Description** | Dettach the logged in cashier from their drawer |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| Drawer | | |
|  | | |
|  | **Usage:** | Drawer(?|n,\_|+Cashier|-Cashier|RESET) |
|  | | |
|  | **Description** | Control cash draws |
|  | | |
|  | **Details:** | ? - drawer associated with current cashier n - explicit drawer number If no parameter - then just open the specified draw +Cashier - Add cashier id to the drawer associations -Cashier - Remove cashier id from the drawer associations RESET - clear the drawer associations |
|  | |
|  | | |
|  | |  |
|  | | |
| Echo | | |
|  | | |
|  | **Usage:** | Echo(Text) |
|  | | |
|  | **Description** | Write to the console, or remote console |
|  | | |
|  | **Details:** | Text - text to write to the console. |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Enumerate | | |
|  | | |
|  | **Usage:** | List=Enumerate(Payment,What) |
|  | | |
|  | **Description** | Return a list of account's subaccounts or a list of transaction accounts. |
|  | | |
|  | **Details:** | Payment gateway ID (see [#Authenticate|Authenticate] for details) OR account id/name if only parameter for listing transaction accounts. Search term or what account to list subaccounts of. |
|  | |
|  | | |
|  | |  |
|  | | |
| Expand | | |
|  | | |
|  | **Usage:** | Expand(ALL|CURRENT,ON|OFF) |
|  | | |
|  | **Description** | Expand (1 item per line) or collapse (like items merged) a transaction or the current part. |
|  | | |
|  | **Details:** | ALL - entire transaction CURRENT - current part ON - expand OFF - collapse |
|  | |
|  | | |
|  | |  |
|  | | |
| File | | |
|  | | |
|  | **Usage:** | File(Target,OFF|ON|TOGGLE|ALL|CURRENT|SELECTED|LAST,Filter) |
|  | | |
|  | **Description** | As [#Print|Print] except prints to a file in the foreign directory |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| FileCompare | | |
|  | | |
|  | **Usage:** | Var=FileCompare(Filename) |
|  | | |
|  | **Description** | Check if two files exist and are the same, returns YES or NO |
|  | | |
|  | **Details:** | Var - return result |
|  | | |
|  | |  |
|  | | |
| FileCount | | |
|  | | |
|  | **Usage:** | Var=FileCount(PathId,Mask) |
|  | | |
|  | **Description** | Return the count of files of mask within the path id given (e.g. Local, Trash etc) |
|  | | |
|  | **Details:** | The path id, as per configuration (Local, Trash, Daily, Queue etc) The file mask to look for e.g. \*.xml |
|  | |
|  | | |
|  | |  |
|  | | |
| FileExists | | |
|  | | |
|  | **Usage:** | Var=FileExists(Filename) |
|  | | |
|  | **Description** | Check if a file exists return YES or NO |
|  | | |
|  | **Details:** | Var - return result |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Finalise | | |
|  | | |
|  | **Usage:** | Finalise(AUTO|NAME|NOSALE|NOTIFY|FORCE|COMPLETE|CONTINUE|SUBTOTAL|CANCEL|INCOMPLETE|TODOS,name) |
|  | |
|  | | |
|  | **Description** | Finalise the transaction, the transaction part or in some other way attempts to finish a transaction |
|  | | |
|  | **Details:** | AUTO - Automatically close the transaction if net value is $0.00 NOSALE - Create and finalise a NOSALE transaction NOTIFY - Create a notification transaction with given name COMPLETE - Complete a transaction, if it is able to be completed SUBTOTAL - Continue a transaction, subtotalling the current part before doing so CONTINUE - Continue a transaction into a new part CANCEL - Cancel a transaction if possible FORCE - Complete a transaction forcibly if required INCOMPLETE - Close a transaction without completing it TODOS - finalise TODO list (and does nothing else), optional name to specify which ones (currently unused) NAME - Name the transaction, for BANKING this sets the comment on the variance name - Optional Name for transaction |
|  | |
|  | | |
|  | |  |
|  | | |
| Flag | | |
|  | | |
|  | **Usage:** | Flag(PRINT|GROUP|SELECT|MARK,ALL|CURRENT|LAST|SELECTED|n,ON|OFF|TOGGLE|value) |
|  | | |
|  | **Description** | Set or unset various flags and values on line items. |
|  | | |
|  | **Details:** | PRINT - Set printed flag GROUP - Set group to value SELECT - Flag items as selected MARK - Flag items as marked ALL - Entire Transaction CURRENT - Current Part LAST - Previous part SELECTED - selected items n - A number of items ON - turn on OFF - turn off TOGGLE - toggle flag value - value for setting GROUP value |
|  | |
|  | | |
|  | |  |
|  | | |
| Hive | | |
|  | | |
|  | **Usage:** | Hive(DOWN|UP|RESEND|PING) or Result=Hive(QUERY,VALUE) |
|  | | |
|  | **Description** | When no return value is specified - Sends commands to the till's hive module |
|  | | |
|  | **Details:** | DOWN - disables the hive UP - enables the hive RESEND - Resends local transactions, tells other tills you have started up. PING - send a ping out to all other tills. When a return value is specified then: QUERY is one of TXSTATUS - returns a global transaction status (as in Tx and Table Picker) for transaciton ID TABLESTATUS - returns a global transaction status for the table specified (VALUE) |
|  | |
|  | | |
|  | |  |
|  | | |
| Hold | | |
|  | | |
|  | **Usage:** | Hold(\_|BROADCAST) |
|  | | |
|  | **Description** | Holds a transaction (or cancels it if empty) saving it, unlocking it, then readying for another transaction |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Identify | | |
|  | | |
|  | **Usage:** | Var=Identify(TableId) |
|  | | |
|  | **Description** | Returns the transaction ID for a given table id, or blank |
|  | | |
|  | **Details:** | Table ID Var will be the TX ID for the given transaction |
|  | |
|  | | |
|  | |  |
|  | | |
| Instruction | | |
|  | | |
|  | **Usage:** | Instruction(?,\*|Qty,Instruction Value) |
|  | | |
|  | **Description** | Set the instructions for selected items |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| Issue | | |
|  | | |
|  | **Usage:** | Var=Issue(InterfaceId,IssuerId,Name,Expiry,Contact,Msg,Value) |
|  | | |
|  | **Description** | Issue or Create an account, coupon, voucher etc on the target customer/interface |
|  | | |
|  | **Details:** | InterfaceId - Interface to create item on (1) SearchTag - Apply search tag, use a - for no search tag. IssuerId - ID used by interface to determine what kind of thing to issue (2) Name - Name of issued item (e.g. customer name, or coupon name) optional account type 'CustomerName/CustomerType' (3) Expiry - expiry from today (e.g. 8d, 2w, 3m, 1y ... days, weeks, months or years). (4) Contact - Contact information to send details to (5) Msg - Text to include in message (6) Value - Value, either currency, integer or blank (7) |
|  | |
|  | | |
|  | |  |
|  | | |
| Layout | | |
|  | | |
|  | **Usage:** | Layout(LayoutId) |
|  | | |
|  | **Description** | Load a new till layout |
|  | | |
|  | **Details:** | LayoutId - name of the layout (no extension) |
|  | | |
|  | |  |
|  | | |
| Limit | | |
|  | | |
|  | **Usage:** | Limit(Amount,HardPercent,WarnPercent) |
|  | | |
|  | **Description** | Sets spending limits on current transaction |
|  | | |
|  | **Details:** | Amount - the amount limit to Hard Percent - the absolute limit as a percentage of the limit Warn Percent - the percent of the limit at which to warn |
|  | |
|  | | |
|  | |  |
|  | | |
| Locate | | |
|  | | |
|  | **Usage:** | Var=Locate(TableId) |
|  | | |
|  | **Description** | Find the location (till id) of a given table |
|  | | |
|  | **Details:** | Table ID Var returns the till ID at which a locked or available till is currently active. |
|  | |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Logging | | |
|  | | |
|  | **Usage:** | Logging(ON|OFF) |
|  | | |
|  | **Description** | Turns on or off command logging |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| Login | | |
|  | | |
|  | **Usage:** | Login(CashierId,\_|[CashierPassword]|Wrist|System) |
|  | | |
|  | **Description** | Login to the till and reload any auto-held cashier transcations belonging to the cashier. |
|  | | |
|  | **Details:** | CashierId to attempt to login No parameter - prompt for password, cashier must have Keyboard enabled in [cashiers.xml] CashierPassword - entered password, cashier must have Keyboard enabled in [cashiers.xml] System - used by internal code to automatically log in and out as required, cashier must have System enabled in [cashiers.xml] Wrist - used when a wrist scanned mask triggers a login even, cashier must have Wrist enabled in [cashiers.xml] |
|  | |
|  | | |
|  | |  |
|  | | |
| Logout | | |
|  | | |
|  | **Usage:** | Logout |
|  | | |
|  | **Description** | Logout of the till and hold any active transaction |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| Lookup | | |
|  | | |
|  | **Usage:** | Var=Position(Choice,List of Choices) |
|  | | |
|  | **Description** | Find the position of choice amoung a list of choices |
|  | | |
|  | **Details:** | Choice - the choice we are looking for List of choices - the list to look in, returns the position of the choice in this list. |
|  | |
|  | | |
|  | |  |
|  | | |
| Macro | | |
|  | | |
|  | **Usage:** | Var=Macro(MacroName|MacroText,TxId|Formatting|\_) |
|  | | |
|  | **Description** | Allows one to evaluate a macro indirectly. |
|  | | |
|  | **Details:** | The macro name that has the name of the macro to evaluate Optional TxId to expand macros in another Transaction Optional Formatting to use for the macro expansion (not supported for other Tx) |
|  | |
|  | | |
|  | |  |
|  | | |
| Mail | | |
|  | | |
|  | **Usage:** | Mail(MessageCode) |
|  | | |
|  | **Description** | Create a message box in the top of the till containing the incoming hive message if one exists; |
|  | | |
|  | **Details:** | MessageCode - the message code to use from Error list |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Mask | | |
|  | | |
|  | **Usage:** | Mask(Maskid|ALL|Prefix,ON|OFF|n) |
|  | | |
|  | **Description** | Enable or disable masks as defined in the [hardware.xml] file. |
|  | | |
|  | **Details:** | Maskid - the mask ID to set enabled or disabled ALL - all masks ON - enabled the mask processing OFF - disable the mask processing n - if the second parameter is an integer, then set a mask Prefix for the next n keyboard lines. |
|  | |
|  | | |
|  | |  |
|  | | |
| Memory | | |
|  | | |
|  | **Usage:** | Memory(Name,Value) |
|  | | |
|  | **Description** | Store a global variable (NOT remembered through till restarts) |
|  | | |
|  | **Details:** | Name - name of variable to store - automatically has an '&' prepended to avoid collisions with significant globals Value - value to store |
|  | |
|  | | |
|  | |  |
|  | | |
| Message | | |
|  | | |
|  | **Usage:** | Message(MessageCode|TillId|CLEAR,Text) |
|  | | |
|  | **Description** | Create a message box in the top centre corner of the till. |
|  | | |
|  | **Details:** | MessageCode - the message code to use from Error list CLEAR - clears a message already on screen, no text required TillId - if an integer is given as the first parameter, till id of till to send text message to. Parameters - parameters to that error |
|  | |
|  | | |
|  | |  |
|  | | |
| Null | | |
|  | | |
|  | **Usage:** | Null |
|  | | |
|  | **Description** | Do nothing, except post command processing |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| Number | | |
|  | | |
|  | **Usage:** | Number(?,Qty) |
|  | | |
|  | **Description** | Adjust the quantity of the current line item by Qty |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Open | | |
|  | | |
|  | **Usage:** | Open(TxId|CashierId|TableId|Tag|TILL,FIRST|LAST|NEXT|PREV) |
|  | | |
|  | **Description** | Opens a transaction, either locally or if unlucked from another till. Fails if no transaction is available. |
|  | | |
|  | **Details:** | TxId - if a 9 character valid transaction ID is given, attempts to load it. TableId - if a valid table number is specified, attempts to load that table CashierId - if a valid cashier id (that isn't also a valid table id) then load a cashier transaction Tag - if a valid tag used by a transaction (note - TILL cannot be a tag here) then load the tagged transaction TILL - Load till transaction FIRST - load the first cashier, tag or TILL transaction LAST - load the last cashier, tag or TILL transaction NEXT - load the next cashier, tag or TILL transaction PREV - load the previous cashier, tag or TILL transaction |
|  | |
|  | | |
|  | | *If a number (e.g. 99) is specified - it first checks if it is a valid table number before checking cashier transactions.* |
|  | | |
| Paste | | |
|  | | |
|  | **Usage:** | Paste(CURRENT|NEW) |
|  | | |
|  | **Description** | Paste the clipboard into the active transaction |
|  | | |
|  | **Details:** | CURRENT - paste into the current part NEW - paste into a new part. |
|  | |
|  | | |
|  | |  |
|  | | |
| Payment | | |
|  | | |
|  | **Usage:** | Payment(Customer.Bucket,Amount|?,Comment|DIRECT) |
|  | | |
|  | **Description** | Same as [#Receipt|Receipt], except direct (not adjusted) amounts. |
|  | | |
|  | **Details:** | If the keyword 'DIRECT' is used, then NO transaction will be created or adjusted and the adjustment will go direct to/from the payment gateway. Note that credit mode intrepretation is REVERSED for Payment command to that of the Receipt command Note credit mode is still respected is still respected in DIRECT mode. |
|  | |
|  | | |
|  | |  |
|  | | |
| Play | | |
|  | | |
|  | **Usage:** | Play(SoundId) |
|  | | |
|  | **Description** | Play the system sound given by the Sound Id |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| Points | | |
|  | | |
|  | **Usage:** | Points(?,\*|Qty,Formula) |
|  | | |
|  | **Description** | Set the loyalty (POINTS) formula, should only be used in behaviours |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| Preview | | |
|  | | |
|  | **Usage:** | Preview(Target,OFF|ON|TOGGLE|ALL|CURRENT|SELECTED|LAST,Filter) |
|  | | |
|  | **Description** | As [#Print|Print] except prints to a preview on screen |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Price | | |
|  | | |
|  | **Usage:** | Price(?,\*|Qty,Formula) |
|  | | |
|  | **Description** | Set the floating price (PRICE) formula, this price is cleared by behaviours - so this command is only useful for behaviours. |
|  | |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| PriceQuery | | |
|  | | |
|  | **Usage:** | PriceQuery(ON|OFF|<blank) |
|  | | |
|  | **Description** | turn on, off or toggle the price query flag (which turns adjustments into price queries |
|  | | |
|  | **Details:** | OFF - normal mode ON - all adjustments become price queries (i.e. how much will the adjustment cost) <empty) - toggles |
|  | |
|  | | |
|  | |  |
|  | | |
| Pricing | | |
|  | | |
|  | **Usage:** | Pricing(Price Level) |
|  | | |
|  | **Description** | Sets the current till (and catalog) pricing level |
|  | | |
|  | **Details:** | Price Level - e.g. Default/2 |
|  | | |
|  | |  |
|  | | |
| Print | | |
|  | | |
|  | **Usage:** | PrinterStatus=Print(Target,OFF|ON|TOGGLE|ALL[\*]|CURRENT[\*]|SELECTED[\*]|LAST[\*]|NOTHING,Filter) |
|  | |
|  | | |
|  | **Description** | Generate a report to the printer based upon what is currently visible on screen. |
|  | | |
|  | **Details:** | Target - The target as defined in [printing.xml] OFF - turn off this report ON - turn on this report TOGGLE - toggle this report STATUS - return a list of offline printers where the target starts with Target ALL - print entire transaction CURRENT - print current part SELECTED - print selected part LAST - print last part NOTHING - print a template with no transaction processing [\*] - optional \* specifies a reprint and to ignore printed flag where relevant (e.g. ALL\*) Filter - macro=value filter for the report |
|  | |
|  | | |
|  | |  |
|  | | |
| Product | | |
|  | | |
|  | **Usage:** | Product(ADD,ProductId,Parameters) |
|  | | |
|  | **Description** | Manipulate the local catalog |
|  | | |
|  | **Details:** | Product Id to maniplulate Paramaters ADD : Name, Groups (GROUP/SUBDEPT/DEPT/CATEGORY), Price |
|  | |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Promise | | |
|  | | |
|  | **Usage:** | Promise(AccountKey,amount) |
|  | | |
|  | **Description** | Adds an account payment promise to the given account key |
|  | | |
|  | **Details:** | AccuuntKey - the existing authenticated account to associate the promise to amount - the positive value to have associated as a pending payment |
|  | |
|  | | |
|  | |  |
|  | | |
| Quantity | | |
|  | | |
|  | **Usage:** | Quantity(Qty) |
|  | | |
|  | **Description** | Set quantity of in the entry area, |
|  | | |
|  | **Details:** | Qty, change the entry from nnnn to nnnn x Qty |
|  | | |
|  | |  |
|  | | |
| Question | | |
|  | | |
|  | **Usage:** | Var=Question(PRODUCT|CUSTOMER|TRANSACTION|INSTRUCTION|STOCK|LIST|PICK|FORCE|DETAIL|FILE|TEXT|EMAIL|URL|PIN|POSTCODE|NUMBER|RANDOM|TABLE||TABLENUMBER|CONTINUE|VOUCHER|TXID,parameters) |
|  | |
|  | | |
|  | **Description** | Ask a question via a popup, returns a Name,Value pair accessible by {Var.1} and {Var.2} or returns a straight {Var} |
|  | |
|  | | |
|  | **Details:** | PRODUCT - Product search, returns id or comma separated ids. Parameters: title, maximum number of products to select, search query CUSTOMER - Search for a customer, parameter: customer type (e.g. Silver, Debtor etc) TRANSACTION - Search for a transaction parameter1: HIVE, INDEX, DAILY, TRASH, LOCAL and parameter2 optional smart search TABLE - Shows a list of tables given an optional smart search. INSTRUCTION - Select instructions, parameter : selector id (from [instructions.xml]) LIST - Select from a list of Name=Value pairs, return {Var.1} and {Var.2} in a pair. PICK - Same as LIST, except don't pop up if there is no or one option - just return it. FORCE - Same as LIST, except force a selection - don't allow a no selection. DETAIL - Show details of Product given the ID, return selected price level for OK, blank for Cancel. FILE - Pick a file from the configured foreign directory. Parameter specifies caption of popup. TEXT - Allow user to enter a line of text with an on screen keyboard, parameter specifies caption of popup. EMAIL - Same as TEXT except enforced a valid EMail address GROUP - Pick a group given a group type ... specific CATALOG for example as parameters to prompt for a CATALOG value. URL - Allow user to verify a URL (or image), returns YES or NO or FAIL. Parameters specify caption and URL IMAGE - Allow user to verify an image, returns YES or NO or FAIL. Parameters specify caption, text and Image URL STOCK - Stock on hand query parameter1 is product id, subsequent parameters optional locations (\* for all) PIN - Allow entry of a P.I.N. - returns PIN or blank POSTCODE - Allow entry of a postcode using the postcodes validation from settings.xml NUMBER - Allows entry of a number - returns number or blank TABLENUMBER - Allows entry of a table, including B for bistro table, returns number e.g. 20, or B20 or blank. RANDOM - Returns a random result either between 1 and n or a and b via RANDOM(n) or RANDOM(a,b) VOUCHER - Gets voucher information from URL, returns FALSE - or the user data defined. CONTINUE - Displays a message, and asks and returns YES or NO. TXID - Asks for a valid TxId specify parameter 1 = tillid to ensure different till or - to allow any, parameter 2 = minites separation of tx to now or - for any. Param 3 is title if the first parameter is TIMED (or the second if the first is DEFAULT=) then the question will respect the configured timeouts/default answers for that question. |
|  | |
|  | |
|  | | |
|  | | *if the first parameter is DEFAULT=<default value>, then that will be passed as the current value, and the actual parameters will start from parameter 3.* |

|  |  |  |
| --- | --- | --- |
| RawPrint | | |
|  | | |
|  | **Usage:** | RawPrint(Target,OFF|ON|TOGGLE|ALL|CURRENT|SELECTED|LAST,Filter) |
|  | | |
|  | **Description** | As [#Print|Print] except prints no formatting characters (for sending to a kitchen printer for example) |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| Recalculate | | |
|  | | |
|  | **Usage:** | Recalculate |
|  | | |
|  | **Description** | Rerun behaviours and calculations on the active transaction |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| Receipt | | |
|  | | |
|  | **Usage:** | Receipt(Customer.Bucket,Amount|?,Comment) |
|  | | |
|  | **Description** | Either receipt or refund (depending on credit mode) |
|  | | |
|  | **Details:** | Customer - Customer as defined by [#Authenticate|Authenticate] or [#Customer|Customer] or [#Card|Card] Bucket - optional bucket id, or similar Amount - Receipt amount, or refund amount, - amount can be Amount+Tip+Surcharge (e.g. 50+10+5) - surcharge in amount can be VENDOR to let the vendor handle the surcharge. ? - the current transaction net value Comment - override the default payment comment |
|  | |
|  | | |
|  | |  |
|  | | |
| Refresh | | |
|  | | |
|  | **Usage:** | Refresh |
|  | | |
|  | **Description** | Refresh the till display |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| Report | | |
|  | | |
|  | **Usage:** | Result=Report(DAILY|SHIFT|POINTS|STOCK|ACCOUNT|PRINT|REPRINT,special...) |
|  | | |
|  | **Description** | Create or reset various reports and totals. |
|  | | |
|  | **Details:** | DAILY - daily report actions SHIFT - shift report actions POINTS - customer points actions STOCK - global stock levels actions ACCOUNT - customer account report via iConnect TillId - Till Id to perform daily report for TILL - the word TILL for daily report on local till DrawerId - drawer number for shift report Customer - customer macro name (from authenticate etc) to do points report for RESET - reset the appropriate report details. (RE)PRINT - save the active report to the reports directory via the provided template and file name |
|  | |
|  | | |
|  | | *Report(DAILY,CREATE,TillId|TILL) or Report(DAILY,RESET,TillId|TILL) Report(SHIFT,DrawerId) or Report(SHIFT,RESET,DrawerId) Report(POINTS,Customer) or Report(STOCK,RESET) REport(ACCOUNT,<accountid>,[<txid>]) - To create an account report Report(ACCOUNT,MERGE) - merge activetransaction into current account report (for final tax invoice printing) Report(ACCOUNT,COMMIT,<accountid>) or Report(ACCOUNT,ROLLBACK,<accountid>) - commit or rollback report Report(PRINT|REPRINT,<TemplateId>,<Filename>)* |

|  |  |  |
| --- | --- | --- |
| Reset | | |
|  | | |
|  | **Usage:** | Reset(?,Qty) |
|  | | |
|  | **Description** | Reset the pricing (including base) on the selected item |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| Retrieve | | |
|  | | |
|  | **Usage:** | Retrieve(tag,CLOSE|\_) |
|  | | |
|  | **Description** | Retrieve all remote open transactions with the given Tag. Optionally attempt to close (AUTO) them |
|  | | |
|  | **Details:** | tag - specify the tag that the transactions should have (e.g. WEDDING) CLOSE - if this parameter is specified, then attempt to close all transactions after they are retrieved PURGE - purge all transactions and remove them from the hive, local directory, do not queue - they should be left in the trash. |
|  | |
|  | | |
|  | |  |
|  | | |
| Reverse | | |
|  | | |
|  | **Usage:** | Reverse(?) |
|  | | |
|  | **Description** | Create a credit mode entry which is the reverse of the selected sale line. |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| ReverseTransaction | | |
|  | | |
|  | **Usage:** | ReverseTransaction |
|  | | |
|  | **Description** | Reverse the current active transaction, or active report - creating a new transaction in the process. |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| RoomCharge | | |
|  | | |
|  | **Usage:** | Roomcharge(Room Id) |
|  | | |
|  | **Description** | Do a room charge to the current room charge system on the server |
|  | | |
|  | **Details:** | Room Id - the full room id and possibly sub id as required by the active room charge system. |
|  | | |
|  | |  |
|  | | |
| RoomIdentify | | |
|  | | |
|  | **Usage:** | Var=RoomIdentify(RoomId) |
|  | | |
|  | **Description** | Call the room interface (via the server) to identify(name) the given room and return that name. |
|  | | |
|  | **Details:** | Room Id appropriate to configured room system |
|  | | |
|  | |  |
|  | | |
| Service | | |
|  | | |
|  | **Usage:** | Service(Command,param1,param2,param3,....) |
|  | | |
|  | **Description** | Send a special request in the transaction to be performed by iConnect or iConnectService. |
|  | | |
|  | **Details:** | Command : Currently only option is EMAIL or NONE (NONE = clears the email settins EMAIL - param1 email PDF template EMAIL - param2 email target address EMAIL - param3 email subject |
|  | |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Shutdown | | |
|  | | |
|  | **Usage:** | Shutdown(\_|RESTART,[\_|FAIL|FORCE) |
|  | | |
|  | **Description** | Shutdown or restart the till software |
|  | | |
|  | **Details:** | No parameter is shutdown RESTART - restart the till UPDATEREMOVE - attempt to retreive new iCell from config server, then restart UPDATELOCAL - attempt to update with file in the update directory |
|  | |
|  | | |
|  | |  |
|  | | |
| Sign | | |
|  | | |
|  | **Usage:** | Sign(CashierId,\_|CashierPassword|Wrist|System,ANY|SECOND|MULTI,Description) |
|  | | |
|  | **Description** | Sign the transaction |
|  | | |
|  | **Details:** | CashierId to attempt to use to sign ANY - allow any cashier to sign, including the cashier or cashiers in the tx SECOND - require the cashier that signs to not be present in the transaction MULTI - allow any cashier to sign, as long as after signing the number of cashiers in the tx is more than one No parameter - prompt for password, cashier must have Keyboard enabled in [cashiers.xml] CashierPassword - entered password, cashier must have Keyboard enabled in [cashiers.xml] System - used by internal code to automatically log in and out as required, cashier must have System enabled in [cashiers.xml] Wrist - used when a wrist scanned mask triggers a login even, cashier must have Wrist enabled in [cashiers.xml] Description - defaults to 'Signed by Joe', or Description if not blank |
|  | |
|  | | |
|  | |  |
|  | | |
| Software | | |
|  | | |
|  | **Usage:** | Software(ICELL) |
|  | | |
|  | **Description** | Perform software updates or other actions |
|  | | |
|  | **Details:** | ICELL Update the ICELL software after icell has notified us of an update |
|  | | |
|  | | *Currently only one option available, may expand in future.* |
|  | | |
| Special | | |
|  | | |
|  | **Usage:** | Special(Payment,Parameters....) |
|  | | |
|  | **Description** | Perform a special function against a payment gateway |
|  | | |
|  | **Details:** | Payment gateway ID (see [#Authenticate|Authenticate] for details) Parameters vary by payment gateway vendor. |
|  | |
|  | | |
|  | |  |
|  | | |
| Split | | |
|  | | |
|  | **Usage:** | Split(Name) |
|  | | |
|  | **Description** | Split the marked items into a new part |
|  | | |
|  | **Details:** | Name - name of new part |
|  | | |
|  | |  |
|  | | |
| SplitBill | | |
|  | | |
|  | **Usage:** | SplitBill(Kind,desc) |
|  | | |
|  | **Description** | n - an integer describing the number of splits to split the bill into |
|  | | |
|  | **Details:** | desc Description to put on the receipt, use a %s to substitute for the changing index/value per description (if appropriate) |
|  | |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Status | | |
|  | | |
|  | **Usage:** | Var=Status(TableId) |
|  | | |
|  | **Description** | Determine the usage status of a table. |
|  | | |
|  | **Details:** | Table Id to query Var returns Locked, Available or Unused |
|  | |
|  | | |
|  | |  |
|  | | |
| Stock | | |
|  | | |
|  | **Usage:** | Var=Stock(?|? x Qty1|ProductList|ProductId x Qty1, Qty2, Pricing) |
|  | | |
|  | **Description** | Same as [#Add|Add] except used when the current customer is a location, rather than a person |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| StockQuery | | |
|  | | |
|  | **Usage:** | Qty=StockQuery(ProductId) |
|  | | |
|  | **Description** | Do a stock query on product id and return how much/many is available at current sell location |
|  | | |
|  | **Details:** | ProductId - product to query Qty - return value for the value (e.g. 1 or 2.77) |
|  | |
|  | | |
|  | |  |
|  | | |
| Store | | |
|  | | |
|  | **Usage:** | Store(Name,Value) |
|  | | |
|  | **Description** | Store a global variable (remembered through till restarts) |
|  | | |
|  | **Details:** | Name - name of variable to store - automatically has an '@' prepended to avoid collisions with significant globals Value - value to store |
|  | |
|  | | |
|  | |  |
|  | | |
| Sundry | | |
|  | | |
|  | **Usage:** | Sundry(Id,Qty,Price) |
|  | | |
|  | **Description** | Add a normal sundry to the receipt, with given price |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |
|  | | |
| System | | |
|  | | |
|  | **Usage:** | System(Operating System command,HIDE|WAIT|HIDE+WAIT) |
|  | | |
|  | **Description** | Execute an operating system command |
|  | | |
|  | **Details:** | The command - cannot include commas HIDE - Minimise the command window WAIT - Wait for the command to complete before continuing. HIDE+WAIT - Both of the above. |
|  | |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Table | | |
|  | | |
|  | **Usage:** | Table(Table Id,<optional note>) |
|  | | |
|  | **Description** | Load, or start a table with given ID. |
|  | | |
|  | **Details:** | Table Id - specifies table type and table number n - Load/create normal table n Bn - Load/create Bistro table n Rn - Reset table number to table n Zn - Load but do not create normal table n NEW - create a new table with first available table number <optional note> - optional hidden note for tables. Defaults to 'Reserved' |
|  | |
|  | | |
|  | |  |
|  | | |
| Tag | | |
|  | | |
|  | **Usage:** | Var=Tag(Transaction Tag) |
|  | | |
|  | **Description** | Sets a transaction tag |
|  | | |
|  | **Details:** | Transaction Tag - the tag to apply, e.g. PARTY etc (assuming Var= is not present) Var - if this syntax is used, retrieves the Tag of either a Transaction ID, or Table Number |
|  | |
|  | | |
|  | |  |
|  | | |
| Tax | | |
|  | | |
|  | **Usage:** | Tax(?,\*|Qty,Formula) |
|  | | |
|  | **Description** | Set the taxation (TAX) formula, should only be used in behaviors |
|  | | |
|  | **Details:** |  |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Timer | | |
|  | | |
|  | **Usage:** | Timer(Name) or Result=Timer(Name) |
|  | | |
|  | **Description** | Start a timer, or get the results of a timer |
|  | | |
|  | **Details:** | Name - the name of the timer to start/stop note that the timer will be cleared on each start Macro=Send(Type,Target,Message,Id) desc Send a mesage to a customer or other party Type : Currently SMS is supported Target : Target of message Message : Text of message Id : Optional message ID note In iCell an error is raised if one occurs In iCaas an attribute is created in the transaction 'SMSResult' with the result of the command. Macro=Post(Method,URL,Username,Password,Response,Name1=Value1,Name2=Value2,....) desc Do a request against a URL or API with parameters Method : Either POST or GET URL : Full base URL of site (excluding parameters) Username: Username if basic auth, blank or '-' if no basic auth Password: Password if basic auth, blank or '-' if no basic auth Response: How to attempt to decode a response JSON:object.object.object......object.attribute TEXT:ALL (returns all text) TEXT:delim.n (returns nth value delimited by delim) delim is 'colon, comma or white' Name=Value: The remaining parameters are the name=Value pairs to send in the request Macro=Crash(ExtendedId,STATUS|PURGE|RESOLVE) desc Perform EFTPOS crash recovery functions ExtendedId : Full extended crash ID (e.g. BCDW0V06U.CDW1S.EFTPOS.P) STATUS : Returns eftpos payment status None : interfaces doesn't support crash recovery Error: Error retrieving status Missing: Payment gateway can't find this transaction Complete: Payment gateway has processed payment Incomplete: Payment gateway has not processed payment COUNT : Count the number of crash flags, always returns a number 0 or greater ExtendedId = ALL - returns total ExtendedId = <valid txid> - returns total number matching that TxiD ExtendedId = <valid paymentid> - returns total number matching that Payment Id ExtendedId = <valid extended id> - returns 0 if not there, 1 if there PURGE : Returns OK if the crash file has been removed from the scratch directory RESOLVE : If the transaction is loaded successfully, this option acts EXZACTLY like a receipt command : including exceptions, if it succeeds without error it will return OK errors : E\_PAYMENT - invalid or missing payment id in crash ID : E\_MISSING - no crash file found in scratch path : E\_INVALID - invalid extended ID : E\_FAILED - failed (to delete scratch record) : E\_TX - filed to load transaction |
|  | |
|  | |
|  | | |
|  | |  |
|  | | |
| Transaction | | |
|  | | |
|  | **Usage:** | Transaction(JSON|XML) |
|  | | |
|  | **Description** | Returns the JSON or XML of a transaction into the console, or to the remote console. |
|  | | |
|  | **Details:** | JSON - format it as JSON XML - Full Finchcorp XML |
|  | |
|  | | |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Trigger | | |
|  | | |
|  | **Usage:** | Trigger(Id,Name,Quantity) |
|  | | |
|  | **Description** | Adds a trigger notify to the transaction |
|  | | |
|  | **Details:** | id - a short alpha code (to be used in behaviours) name - a descriptive name to be displayed on the transaction qty - a quantity of the trigger (for testing in the behaviours) |
|  | |
|  | | |
|  | |  |
|  | | |
| UnIssue | | |
|  | | |
|  | **Usage:** | Unissue(Interface,IssuedId) |
|  | | |
|  | **Description** | Unissue, void or cancel an acoount, coupon, voucher etc. |
|  | | |
|  | **Details:** | InterfaceId - Interface to create item on IssuedId - id of issued item |
|  | |
|  | | |
|  | |  |
|  | | |
| Value | | |
|  | | |
|  | **Usage:** | Var=Value(Param1) |
|  | | |
|  | **Description** | Set var to the value given |
|  | | |
|  | **Details:** | Param1 is evaluated and stored in Var |
|  | | |
|  | |  |
|  | | |
| View | | |
|  | | |
|  | **Usage:** | View(TxId|CURRENT) |
|  | | |
|  | **Description** | View a completed transaction or return to the active transaction |
|  | | |
|  | **Details:** | TxId - 9 character transaction ID of a closed transaction REPORT - views the current 'report' transaction CURRENT - returns view to current active transaction |
|  | |
|  | | |
|  | |  |
|  | | |
| Void | | |
|  | | |
|  | **Usage:** | Void(MONEY|PRODUCT|STOCK|?) |
|  | | |
|  | **Description** | Void or remove completely the entry selected (set quantity and value to zero, then remove) |
|  | | |
|  | **Details:** | MONEY - only void if the selected item is money PRODUCT - only void if the selected item is product STOCK - only void if the selected item is stock ? - void without concern for what the item is |
|  | |
|  | | |
|  | |  |
|  | | |
| Window | | |
|  | | |
|  | **Usage:** | Window(Appname, Action) |
|  | | |
|  | **Description** | Controls the window of a windows application |
|  | | |
|  | **Details:** | AppName - name of app (e.g. notepad) Action - one of MAXIMIZE, MINIMIZE, RESTORE, SHOW, HIDE or blank - blank restores the window to its normal location. |
|  | |
|  | | |
|  | |  |