iCell Macro Guide

# INTRODUCTION

Various parts of iCell allow the use of Macros. Macros are a simple text substitution that allow general scripts to be written and have values filled in later.

A macro generally works in the following way (with variations documented in this guide):

* A macro will have a name for example *cashier*
* A macro will have a value for example *John Smith*
* A script will refer to the macro like this: *{cashier}*
* “*The cashier is {cashier}*” would be expanded to “*The cashier is John Smith*”.

Macros can be used in four places, however macros in Behaviours are covered in the behaviours documentation and parameter macros are covered in the button scripting manual. This guide covers built in macros and their use. This guide therefore covers macros used in:

* Button scripts (buttons.txt) – excluding user macros and parameter macros
* Report templates (templates.txt)
* Receipt Display Fields (fields.xml)

Macros in iCell frequently have compound names which determine where to target a macro, for example:

* {t.macro} – is a macro that returns a transaction value (e.g. end date)
* {tx.macro} -> is a macro that returns a total, or list of values from all details (e.g. quantity of items)
* {m.macro} -> is a macro that returns a value from a detail (in this case money e.g. value), detail macros can only be used when a detail is known such as in the field definitions, or in a detail loop in a report template.

Some macros that are numeric (e.g. quantities and values) can be evaluated in three different ways (triple macros) depending on what you surround the value with:

* {normal} – curly braces mean evaluate the macro as is
* [positive] – square brackets mean always evaluate the macro as a positive value.
* (reverse) – round brackets mean always evaluate the macro using the opposite sign.
* Triple macros can also have a format specified at the end such as {x,%8.2f} or {x,6}

Macros are in four main groups:

* Processed by the transaction, and thus usable everywhere.
* Processed by the **print manager** and thus only available in the report templates
* Processed by the **transaction manager** and thus only available in the button scripts.
* Processed in other places (e.g. accounts, catalog etc)

In addition – some macros can return a ‘do not print’ flag, which signals to the reporting template not to print any row containing such a value (useful for omitting rows that have no data).

# Transcation manager macros

**Transaction Manager** macros can only be used in button scripts and return values useful for decision making in those scripts. The following is a list of those macros.

* *{cashierId}* – Current till cashier ID
* *{cashierdrawer}* – current till cashier ID assigned drawer
* *{cashiersecurity}* – current till cashier security string
* *{cashiertoken}* – token used by cashier to log in
* *{cashier}* – current till cashier name
* *{date}* – current date (dd-mmm-yyyy)
* *{date*.year} – current year
* *{date*.month} – current month (1-12)
* *{date*.day} – current day of month (1-31)
* *{date*.dow} – current day of week (Monday, Thursday etc)
* *{time}* – current time (hh:mmAM/PM)
* *{time*.hour} – current hour (0-23)
* *{time*.minute} - current minute (0-59)
* *{system*.build} – icell Build number
* *{system*.version} – full icell version
* *{hivetime}* – string representing current hive time (ALPHA)
* *{hivetables}* – how many tables are visible on the hive
* *{hiveoldtables}* – how many hive tables are old (based on setting in settings.xml)
* *{status}* – OPEN or CLOSED status of current transaction
* *{onhold}* – quantity of transactions on hold
* *{connected}* – YES or NO if hive connected
* *{creditmode}* – current credit mode (on, sale, item, off)
* *{usbstatus}* – status of USB backup drive (YES for plugged in, NO for error, or blank if missing)
* *{usberror}* – error message from USB if in error state
* *{till*.function} – normal till or menu board till
* *{onhold}* – how many transactions (including active one) are on local till
* *{toprint}* – how many items are yet to print
* *{haveprinted}* – how many items have been printed.

**Transaction Manager** macro processing also processes the following macros:

* Settings Macros (abn, company, venue, greeting and address)
* **Hive Macros** {@hive.macro}
* Client Macros (configured locally expanded macros – clientmacros.ini)
* Global Storage Macros (permanent user defined macros – scratch/storage.xml)
* **Catalog Macros** {@Catalog.productid.macro}
* **Account Macros** for Current Customer {@.macro}
* **Transaction Macros** for Active Transaction {t.macro} and {tx.macro}
* Payment Gateway Macros **Account Macros** {@interface.a.macro}
* Local Money Storage Macros {money.macro}
* **Transaction Macros** for Selected Part Only {sx.macro}

# PRINT MANAGER

The print manager is responsible for creating output to the receipt printer, the details of the scripting system for that are described in the iCell Report Template Guide.

The following is a list of macros available only to the reporting templates:

* {reported} – unique ID for this output
* {template\_group} – group defined in the definition in printing.xml
* {template\_groupvalues} – group values defined in the printing.xml
* {mergetime} – yes if common items at different times are merged
* {mergecashier} – yes if common items sold by different cashiers are merged
* {mergepricing} – yes if common items with different pricing are merged
* {reprint} – print already printed items.
* {cashierid} – active cashier id
* {cashierdrawer} – active cashier drawer id
* {cashier} – active cashier name
* {customerid} – active customer id
* {customertype} – active customer type
* {customer} – active customer name
* {flagged} – set in the printing.xml
* {date} – friendly date dd-mmm-yyyy
* {time} – friendly time hh:mmAM/PM
* {datetime} – same as {date} {time}
* {timestamp} – yyyy-mm-dd hh:nn:ss
* {qty} – in a product loop the quantity of items
* {count} – in a product or count loop the quantity of lines
* {group} – in an each or group loop, specifies the item in the each loop you are up to.
* {iqty} – in an instruction loop, the quantity of the item
* {zqty} – in an instruction loop, the quantity of the item if more than 1.
* {iname:n} – in an instruction loop display instruction in one of 12 formats (see INSTRUCTION LOOP FORMATS in reference section)

**Print Manager** macro processing also processes the following macros:

* **Catalog Macros** *{@Catalog.productid.macro}*
* Global Storage Macros *{@macro}* (permanent user defined macros – scratch/storage.xml)
* **Local Configuration Macros**
* Settings Macros (abn, company, venue, greeting and address)
* **Transaction Macros** for Active Transaction *{t.macro}* and *{tx.macro}*
* **Transaction Macros** for Active Detail *{m.macro}* etc if in a detail loop.

Hive Macros  
There are two basic hive macros of the form {@Hive.macro}, the first does stock level checks on the hive and the second does tag counts on the hive.

Stock level checks are done as follows:

* {@Hive.productid} – returns either ‘Blank’ or the quantity of stock in the hive
* {@Hive.productid.threshold} – returns either NoStock, LowStock (if under threshold) or Available (if above threshold)

The tag count is done as follows:

[{@Hive.tag.tagname}](mailto:%7b@Hive.tag.tagname%7d) – returns the number of transactions with that tag in the hive.

Catalog Macros  
Catalog macros are of the form {@Catalog.productid.macro} or {@Catalog.barcodetype.productid}

There are three types of barcode functions, examples:

The first creates an expiring barcode of the form VC<productid><expiry> for use in a printing template:

* {@Catalog.VC.1234.18} - 18 days from now
* {@Catalog.VC.1234.25/10/2018) - specific date ...

The second creates an expiring QRCode of the form QRCODE:<productid><expiry> for use in a printing template:

* {@Catalog.QR.1234.18} - 18 days from now
* {@Catalog.QR.1234.25/10/2018) - specific date ...

The third returns the expiry date (dd mm yyyy) of the a barcode that would be created by the same macro but with VC or QR. As follows:

* {@Catalog.XP.1234.18} - 18 days from now
* {@Catalog.XP.1234.25/10/2018) - specific date ...

The second kind of Catalog macro retrieves a value from a specified product, e.g. {@Catalog.product.macro} will return the value of macro for product id given. The macros you can retrieve are:

* {id} – product id
* {name} – product name
* {longname} – long name
* {buttonname} – button name
* {description} – product long description
* {parent} – product id of parent product
* {menu.name} – get menu pricing for name value.
* {price.name} – get price for price level given by name
* {group.groupname} – get group value for product’s group name
* {flag.flagname} – YES or NO depending on value of flag name of product.

Local Config Macros

The config macros are of the form {CONFIG.Macro} and are listed here:

* {tillid} – configured till id for the till (not the transaction)
* {fulltillid} – configured text based till id
* {configserver} – configured config server
* {cellid} – configured cell id
* {locationid} – configured location id for till (not transaction}
* {till} or {tillname} – configured till name
* {location} or {locationname} – configured till location name
* {machineid} – configured machine id used to identify machine to config server
* {printername} – default printer name
* {windowed} – YES if configured to run in window, NO otherwise
* {editor} – YES if editor is enabled
* {dualmonitors} – YES if dual monitors configured
* {hiveenabled} – YES if hive is enabled
* {hiveport} – hive broadcast port
* {hivedirect} – hive direct port (incoming)
* {hivepassword} – hive direct password
* {hivehttp} – hive http (not used on iCell)
* {hivegroup} – broadcast group address
* {hivebind} – local IP address for iCell to bind hive to
* {hiveserver} – hive direct address for server (ics)
* {hivetimeout} – time out on hive
* {databasepath} – configured database path (not used in icell)
* {databaseschema} – configured database schema (not used in icell)
* {databasevendor} – configured database vendor (always MS) (not used in iCell)
* {databasevendorlib} – optional DLL name for database vendor (not used in iCell)
* {databaseuser} – login name for database (not used in iCell)
* {databasepasswd} – login password (not used in iCell)
* {databaseport} – port to connect to database on (not used in iCell)
* {databasetimeout} – timeout for connections to database (not used in iCell)
* {databaseliveupdates} – refresh updates from database {not used in icell}
* {ftpserver} – configured FTP server {not used in icell}
* {ftpuser} – ftp login name (not used in icell)
* {ftppassword} – ftp login password (not used in icell)
* {ftppassive} – toggle ftp passive mode (not used in icell)
* {ftpport} – ftp port to use (not used in icell)
* {pathXXXX} – the path for specified directory (e.g. {pathLocal})
* {urlXXXX} – same as {pathXXXX} but specified in URL format
* {fileXXXX} – configured file name for file (e.g. {fileButtons}
* {logXXXX} – YES or NO for given log file enabled {e.g. lognormal}

Account Macros  
These macros will be prefixed with either the account name, or @ for current account, or ‘a’ when using a for each account loop in report templates., For example {@.name} would refer to the current account customer name.

Where relevant these macros support do not print and triple macro expansion.

* *{id}* – Account Id (internal reference}
* *{cardid}* – Account card id/swipe id
* *{pricing}* – customer pricing overrides
* *{mainid}* – Primary key from external system.
* *{name}* – Main name for account
* *{subid}* – secondary or sub id for account (e.g. folio id)
* *{subname}* – secondary or sub name for account (e.g. folio name)
* *{interface}* – interface id
* *{type}* – member type
* *{rating}* – account/customer rating
* *{email}* – account email address
* *{address}* – account physical address
* *{phone}* – account telephone
* *{expiry}* – account expiry date, dd-mmm-yyyy
* *{commence}* – account commencement date, dd-mmm-yyyy
* *{valid}* – YES or NO
* *{requested}* – Last requested amount
* *{approved}* – Last approved amount
* *{adjustment}* – Last adjustment
* *{extra}* – Last extra (e.g. cashout)
* *{limit}* – Account limit
* *{balance}* – Account balance
* *{owes}* – Current owed
* *{tip}* – Last tip
* *{surcharge}* – Last surcharge
* *{cash}* – cash balance
* *{points}* – points (via exchange rate) balance
* *{rawpoints}* – raw points balance
* *{cashout}* – last cashout request
* *{value}* – Last request product value sent
* *{count}* –Current coupon usage count
* *{frequency}* – Current coupon usage frequency
* *{bucket}* – interface specific balance of bucket

Transaction Macros  
 Transaction

Where relevant these macros support do not print and triple macro expansion.

* *{t.sdate}*- Start Date (dd-mmm-yyyy)
* *{t.edate}* – End Date (dd-mmm-yyyy)
* *{t*.*stime} –* Start Time (hh:mmAM/PM)
* *{t.etime}* – End time (hh:mmAM/PM)
* *{t.cashierid}* – Transaction Cashier ID
* *{t.tillid}* – Transaction active till id.
* *{t.id}* – Transaction ID
* *{t.net}* – Current transaction net value
* *{t.covers}* – Covers count
* *{t.value}* – Transaction product value
* *{t.paid}* – Current total money value (amount paid)
* *{t.limit}* – Current normal limit on transaction
* *{t.hlimit}* – Hard limit
* *{t.wlimit}* – Warning limit
* *{t.alimit}* – Current transaction available limit (Limit + Net)
* *{t.change}* – Total change for all parts in transaction.
* *{t.table}* – Table type either BISTRO or NORMAL
* {*ta.AttributeName}* – Current value of attribute assigned by attribute command.
* *{accountname.macro}* – Transaction account macro 🡪 see **Account macros**
* *{tx.macro}* – Totalling Current Part Macros (normal) 🡪 see **Transaction Part Macros**
* *{cx.macro}* – Totalling Current Part Macros (cancelled) 🡪 see **Transaction Part Macros**

Part

A part macro is normally called using {sx.macro} for selected, or {tx.macro} for current.

Where relevant these macros support do not print and triple macro expansion.

The macros are:

* *{location}* – Location for part
* *{locatoinname}* - same as {location}
* *{locationid}* – Location ID for part
* *{tillid}* – Till ID on part
* *{tillname}* – Till Name on part
* *{cashiered}* – Cashier Id on part
* *{date}* – Date of part (dd-mmm-yyyy)
* *{time}* – Time of part(hh:mmAM/PM)
* *{txid}* – Transaction Id of transaction owner of this part.
* *{txclass}* – transaction class of transaction owner of this part.
* *{receiptid}* – Combination of Txid and Part Id
* *{table}* – Current tableid, or ‘none’ if no table assigned.
* *{gp}* – total GP% across this part.
* *{split}* – has a split (YES)
* *{splitvalue}* – value of split or 0 if no split
* *{p*.*macro} –* Evaluate product macro🡪 see **Transaction Detail Macros**
* *{s.macro}* – Evaluate stock macro 🡪 see **Transaction Detail Macros**
* *{c.macro}* – Evaluate customer macro 🡪 see **Transaction Detail Macros**
* *{m.macro}* – Evaluate money macro 🡪 see **Transaction Detail Macros**
* *{?.macro}* – Evaluate autodetect(M,P,S) macro 🡪 see **Transaction Detail Macros**
* *{v.type.method}* – Evaluate value of macro 🡪 see **Transaction Detail Macros**
* *{q.type.method}* – Evaluate quantity of macro 🡪 see **Transaction Detail Macros**
* *{u.macro}* – Evaluate user/cashier macros 🡪 see **Transaction Detail Macros**
* *{b.macro}* – Evaluate behaviour macros 🡪 see **Transaction Detail Macros**
* *{g.macro}* – Evaluate group macros 🡪 see **Transaction Detail Macros**

Detail

A detail macro is most frequently called like {tx.detailmacro} to give a sum or list of all the values in all the details. However it can be called as just {detailmacro} in the fields definition and within detail loops in report templates.

Where relevant these macros support do not print and triple macro expansion.

A {detailmacro} is of the form {x.macro} where x is one of the following:

* M – a money detail
* P – a product detail
* S – a stock detail
* D – a local money detail
* L – a local product detail
* N – a notify detail
* ? – auto detect from the above
* U – a detail cashier/user macro
* B – a detail behaviour macro
* Q – a quantity macro
* V – a value macro
* G – a group macro
* A – a detail account macro – see Account Macros

The following is a list of the normal macros for M,P,S,D,L,N, U and B (e.g. {m.marked}).

In detail macros, the (A,B,C..) at the end of each line represents the types of macro this is valid for. For example {tx.c.rating} is valid, but {tx.p.rating} is not.

* *{name}* – Name of detail (P,S,M,N,C,U,B)
* *{count}* – either 0 or 1 – used via {tx.p.count} etc to count items. (P,S,L,M,N,D)
* *{drawer}* – Drawer Id of detail (U)
* *{id}* – ID of detail (P,S,M,N,U,T,B)
* *{kind*} – Friendly name for single letter type (e.g. P -> Product, U -> Cashier) (All)
* *{time} – detail time as hh:mmAM/PM* (All)
* *{date} – detail date as dd-mmm-yyyy* (All)
* *{tabcharged} – YES or NO depending on if detail is tab charged (All)*
* *{partlocation}* – location of detail’s part (All)
* *{parttill} -*  till of detail’s part (All)
* *{partlocationid} –*location ID of detail’s part (All)
* *{parttillid} –*till id of detail’s part (All)
* *{paymentmethod}* – Payment method of detail (M,N,D)
* *{paymentname}* – Payment name of detail (M,N,D)
* *{paymentid}* – Payment Id of detail (M,N,D)
* *{paymentbin}*– Payment bin of detail (M,N,D)
* *{comments}* – Comments field of detail (All)
* *{instructions}* – display friendly list of instructions for detail (All)
* *{htmlinstructions}* – HTML friendly list of instructions for detail (All)
* *{marked}* – ‘\*’ or ‘’ depending on if marked (All)
* *{points}* – either paid points (M) or points value (P)
* *{value}* – value of detail line (P,S,N,M)
* *{type}* – Customer type (C), Detail Type (P,S,M,N,D,L)
* *{rating}* – Customer rating (C)
* *{qty}* – Quantity on detail line (P,S,N,M)
* *{direction}* – ‘+’ or ‘-‘ depending on stock movement (P,S)
* *{extax}* – either ‘\*’ or ‘’ depending on if an ex tax line (P,S,L)
* *{printed}* – either ‘\*’ or ‘’ depending on if detail line printed (P,S,L)
* *{accref}* – account reference for detail (M,N,D)
* *{payref}* – payment reference for detail (M,N,D)
* *{paytype}* – payment type (M,D)
* *{payrecord}* – payment record (M,D)
* *{normalcost}* – normal cost of item (P,S,L)
* *{normalsell}* – normal sell of item (P,S,L)
* *{itemgp}* – Percent item margin (P,S,L)
* *{itemmargin}* – Dollar margin on line (P,S,L)
* *{profit}* – Profit on line item (P,S,L)
* *{itemcost}* – same as {productcost} (P,S,L)
* *{itemvalue}* **–** sell price + instruction value (no quantity) (P,S,L)
* *{linemargin}* – line margin by qty (P,S,L)
* *{linecost}* – line cost by qty (P,S,L)
* *{sell}* – actual sell of line by qty (P,S,L)
* *{savings} –* customer savings on line (P,S,L)
* *{surcharge}* – customer surcharge on line (P,S,L)
* *{discount}* – discount on line (P,S,L)
* *{tax}* – tax on line (P,S,L)
* *{nsd}*– negative savings/discount (P,S,L)
* *{psd}*– positive savings/discount (P,S,L)
* *{groupname}* – returns group value, e.g. DEPT might return BEER (G)

The macros for Q and V are somewhat different.

Normal Q/V mode: they use the format {q or v.type.method} for example {tx.q.sale.1245)

* Type is one of the following: Sale, Return, Change, Sundry, Refund, Receipt, Rounding, CashoutOf, CashoutFrom, Cashout, Stockin, StockOut, Accrual, UnAccrual, Deposit, Withdraw, Taxation, Cover, Subtotal, Trigger, Note, Nothing
* Method is either Payment Method (for money detail lines) or Product ID for other details.
* If Type is ‘\*’ then All types match
* If Method is ‘\*’ then all methods match
* An optional @Flag is a test against payment flags (configured in payment interfaces), if it is specified and doesn’t match it isn’t counted.
* Example [{tx.q.refund@cash}](mailto:%7btx.q.refund@cash%7d).

Group Q/V mode: they use the format {q or v.groupname.groupvalue} for example {tx.q.dept.beer).

The Q/V mode macros are generally used to calculate tx wide amounts.

The ‘A’ macro gets the value from ACCOUNT MACROS for the customer associated with the detail (e.g. tx.a.type).

Simple addition and subtraction can be done within transaction detail macros for example {tx.p.value-p.cost}, notice that the ‘tx’ part is not repeated.

# SPECIAL MACROS

There are two categories of macros used which are outside the normal macro system:

* Configuration Server macros (replaced before sending to iCell)
* Live update macros (which are limited due to performance)

## LIVE UPDATE MACROS

Live update macros can be used in button text or button bubbles (the red circle in the bottom right). These macros assume that the button has the correct ITEM id (e.g. item id for product, table id for table button)

These macros are as follows:

* {price} - price of item (for Item buttons)
* {name} - name of item (for Item buttons)
* {avail} - quantity available via hive (for Item buttons)
* {qty} - quantity via local stock file (for Item buttons)
* {tag} - count of transactions with this tag
* {cashier} – current cashier name
* {tablebalance} - owed on table (for table buttons)
* {tablename} - name of table (or blank) (for table buttons)
* {tablestatus} - status of table (for table buttons)
* {=macro} - persistance macro (for captions only)

## CONFIG SERVER MACROS

These macros will occur in the templates for some config files (in cfTemplates).

### Machine level macros

* {config.machine.systime} – current server time when config was requested
* {config.machine.MID} – machine ID of machine config is served for.
* {config.machine.TILLID} – till id
* {config.machine.TILLNAME} – till name
* {config.machine.LOCID} – location id
* {config.machine.LOCNAME} – location name
* {config.machine.DIRIP} – direct hive IP address
* {config.machine.DIRPORT} – direct hive port
* {config.machine.BCASTIP} – broadcast IP address
* {config.machine.BCASTBIND} – broadcast bind address
* {config.machine.BCASTPORT} – broadcast port
* {config.machine.SERVERIP} – server hive direct address
* {config.machine.SERVERPORT} – server hive direct port
* {config.machine.HIVEPASSWD} – hive password
* {config.machine.HIVETIMEOUT} – hive timeout
* {config.machine.HTTPTIMEOUT} – http timeout
* {config.machine.SoftwareUpdate} – Y or N if software update available
* {config.machine.RKEY} – Registration key
* {config.machine.SKEY} – Registration serial number

Configuration macros are of the form {cfg.Config Key} and return the value that the configuration system sees for that configuration key (via cfDefaults and cfMacros tables).

### Special conditional macros:

* **{cfg.IfYes. <value 1>=<value 2>}**Returns YES or NO depending on if value 1 and value 2 are equal
* **{cfg.YesIf.<yes or no>?<yes value>:<no value>}**Returns either yes value or no value depending on if <yes or no> value
* **{cfg.ChooseFrom.<index>?<value 1>,<value 2>,<value 3>}**Returns one of the values depending on the index

Notes:

* The < and > are NOT to be included, they are just representing different values that can be used.
* The <values> above can be other cfg. macros or plain test values.
* Be aware of complications if there are =,?,: or , in these <values>.
* The default if <index> for Choosefrom is <value 1> if Choosefrom is not a number or is lower than 1.
* If <index> is a number for which there is no <value> then a blank is used.

Examples:

* {cfg.ChooseFrom.2 ?Fred,Barney,Wilma,Betty} = Barney (Returns 2nd value in the list)
* {cfg.ChooseFrom.bob ?Fred,Barney,Wilma,Betty} = Fred (returns 1st value when not a number)
* {cfg.ChooseFrom.0 ?Fred,Barney,Wilma,Betty} = Fred (returns 1st value when less than 1)
* {cfg.ChooseFrom.-1 ?Fred,Barney,Wilma,Betty} = Fred (returns 1st value when less than 1)
* {cfg.ChooseFrom.6 ?Fred,Barney,Wilma,Betty} = <blank> (returns blank value when greater than range)

### Loop macros

Loop macros are defined on a LINE level within a template file as in the following example:

<--#EACH FINCHCORP Member Types--><setting name="membertype.{@@GRID.ID}" value="{@@GRID.MemberType}"/>

Loop macros must start the line with literally “<--#EACH *a config key*-->“, ‘a config key’ will be the key name in cfDefaults/cfMacros of a grid configuration item.

The grid is retrieved as a list of rows with a value for each field/column in the row.

The remainder of the line will use the {@@GRID.gridcolumn} macro to refer to the value of the row for that column.

The macro will then generate a line for EACH row of the grid, replacing the macros with the column values for each row as defined using the @@GRID macros.

The above might output something like the following:

<setting name="membertype.DEFAULT" value="Loyalty"/>  
<setting name="membertype.1000" value="Silver"/>  
<setting name="membertype.2000" value="Gold"/>  
<setting name="membertype.3000" value="Platinum"/>

# REFERENCE

## INSTRUCTION LOOP FORMATS

When using the {iname} macros for instruction loops, the following formats (specified as a number from 1 to 12 or blank) are available, using ‘MEDIUM WELL’ as the instruction name here.

* blank = MEDIUM WELL
* 1 = (1) MEDIUM WELL  
   (2) MEDIUM WELL
* 2 = MEDIUM WELL  
   (2) MEDIUM WELL
* 3 = MEDIUM WELL (1)  
   MEDIUM WELL (2)
* 4 = MEDIUM WELL  
   MEDIUM WELL (2)
* 5 = SGL MEDIUM WELL  
   DBL MEDIUM WELL
* 6 = MEDIUM WELL  
   DBL MEDIUM WELL
* 7 = MEDIUM WELL (SGL)  
   MEDIUM WELL (DBL)
* 8 = MEDIUM WELL  
   MEDIUM WELL (DBL)
* 9 = (x1) MEDIUM WELL  
   (x2) MEDIUM WELL
* 10 = MEDIUM WELL  
   (x2) MEDIUM WELL
* 11 = MEDIUM WELL (x1)  
   MEDIUM WELL (x2)
* 12 = MEDIUM WELL  
   MEDIUM WELL (x2)
* $n = above format (n = 1 to 12) - but with currency value appended if non-zero