iCell Behaviours Guide

# INTRODUCTION

Behaviours are stored in a plain text file (normally behaviours.txt) and consist of several behaviours in the following format:

1. # - full line Comments
2. // partial line comments (only on behaviour command lines)
3. NAME – sets the name of the behaviour
4. OPTIONS – sets various options for the behaviour
5. BEHAVIOUR commands – the core of the behaviour
6. OR – allows for optional additional sets of BEHAVIOUR commands if the previous set fails.
7. END commands – specifies the end of the behaviour, or the section.

The NAME command starts off a behaviour, and sets the name of the behaviour (to be used in the local behaviours). It also specifies the parameters of the behaviour.

For example:

Name ICTimedPromo(DateStart,DateEnd,TimeS,TimeE,DOW,PromoID,Price,Priority)

Specifies a behaviour with name ‘ICTimedPromo and 8 parameters. These parameters are simple replacement macros for the remainder of the behaviour.

The END commands finish off either an entire behaviour, or a section of an existing behaviour. The difference between the two, is that after and END you need to specify a new name and options for another behaviour, but if you use the OR command it allows you to specify one or more alternate sections for the behaviour.

OPTIONS commands are as follows and specify various configurations for the behaviour:

* TEXT – specifies the text to display in the receipt and transaction for this behaviour (e.g. Free Coke Special)
* PRIORITY – specifies the priority for adjustments caused by this behaviour that don’t have priorities based on price levels. Priorities are numbers from 0 to 99, with 99 being the highest. A lower priority adjustment cannot overwrite a higher priority adjustment.
* DEBUG – this option with no parameters specifies that debug logging should occur for this behaviour.
* SORTING – One of UnSorted, CheapestFirst, ExpensiveFirst, EarliestFirst, LatestFirst. This specifies the order in which matches for **conditions** should be made against items. For example ‘LatestFirst’ means that the condition will match those items first.
* OPTION – One of Priority, Cheapest, Dearest. This specifies the order in which **adjustments** should be made, the default of Priority uses the priority specified above – the others ignore the priority and adjust based upon cheapest or dearest.
* TIMING – one of Auto, Before and After. Determines the phase during which the behaviours are executed. Before means execute first, after means execute AFTER the first behaviours are run AND their adjustments made. Auto uses the default - Tax and Loyalty behaviours use After, others use Before.

BEHAVIOUR commands actual perform actions or tests on the transaction and form the core of the behaviour system. The commands are:

* CONDITION – specifies a condition under which the behaviour should continue to operate. Conditions are complex and will be discussed in detail in Condition Section of this document.
* MODIFY – specifies an adjustment to make, these adjustments are almost exactly the same adjustments you can make from a command button. This will be discussed in the ADJUSTMENTS section of this document.
* SET – set a macro in the behaviour. This includes being able to do simple maths, for example SET COUNT={macro}\*4, or SET MAX=>4,5,2,10 will set MAX to the largest of the list.  
  SET X = N CHOOSE A,B,C,D will set X to the Nth item (with 0 giving the first item) in the list defaulting to the last if N is an integer, otherwise SET X = N CHOOSE A=1,B=2,C=3,DEFAULT=4 will set X to the value which matches the name N.
* TODO – inserts a todo list entry into the transaction which will be executed upon completion of the part. This will be discussed in the TODO section of this document.
* QUESTION – asks a question of the user (or a few other special functions), and caches the result for the duration of the transaction part. This will be discussed in the QUESTION section of this document.
* ABORT – aborts the behaviour execution and the adjustment that initiated it.

Behaviour command conditional execution:

?Value BEHAVIOUR COMMAND

Will skip the execution of behaviour command if Value is NO, OFF, FALSE, 0 or empty. A typical use would be for the abort command

?{?1} ABORT Not allowed

# CONDITIONS

Conditions have multiple parts to specify basically two things – what to test, and what to test it again. The behaviour section stops if any condition returns 0.

The syntax of a CONDITION is as follows:

CONDITION [BATCH x/y] [CONTINUE|STOP:ON|STOP:OFF|STOP:YES|STOP:NO] [NOT] [ABS] <TestVar> [OF <TestItem>] [FOR <TestGroup>(<TestTarget>)] [Optional Per test] <TestOper> <TestValue>

Basically this consists of:

* Optional Batch specifier with x=id of this condition, y being the maximum number of conditions to evaluate. A condition is skipped and processing continues if the batch id exceeds the maximum See Batch information in adjustments section for more information.
* Optional ‘NOT’ specifies to reverse the condition
* Optional ‘ABS’ specifies to make all values positive before testing – except your test value.
* Optional ‘CONTINUE’ specifies that the behaviour should continue execution always   
  Optionally use ‘STOP:OFF’ or ‘STOP:NO’ to replicate the CONTINUE function  
  Optionally use ‘STOP:ON’ or ‘STOP:YES’ to replicate normal function
* Required Test variable (i.e. what we are testing)
* Optional ‘OF’ test item specifier, defaults to ‘Till’, see below for more details.
* Optional ‘FOR’ test group specifier, defaults to ‘All’, see below for more details.
* Required test operator, see below for details.
* Required test value (i.e. what we are testing against).

A comprehensive list of each of these:

Test variables:

* Always – matches anything
* Pricing – product pricing
* Customer – customer id
* CustType - customer type
* CustRating – Customer rating
* User – Logged in cashier
* Table – Current table number
* Quantity – some quantity of product
* Value – some value of product
* Time – date or time
* Literal – literal comparison of a given value with another
* Till – till id
* Location – location id

Test Items (not all of these are implemented and tested yet):

* Receipt – current part of the transaction
* Till – the till itself
* Location – the venue/location (not implemented)
* Detail – specific detail

Test Groups (used for quantity and value test variables)

* Product – a product or list of products (specified by the testtarget e.g. 1234)
* Group – a catalog groupname/groupvalue (specified by the testtarget e.g. DEPT:BEER)
* NoGroup – not in a catalog groupname/value (see group)
* Customer – the customer associated with the test item (specified as customer id)
* Trigger – Especially targets trigger items by trigger name, rather than product detail.
* User – the cashier associated with the test item.
* Flag – As Group, but just presence of catalog flag
* NoFlag – Missing given flag
* Answer – As Flag, except tests the behaviour answer as YES
* NoAnswer – As NoFlag, except tests the behaviour answer as NO
* Value – used with Literal to specify the literal value to test
* All – the default, meaning match all details.

Optional [Optional Per test] section

* PER – prefixes the per test section
* <testgroup> - currently only GROUP is suppored
* (test value) – the group name to test e.g. (DEPT)
* <per oper> - one of ‘OF’, ‘UNDER’ or ‘OVER’
* <per amount> - either a value or quantity depending on the condition test variable
* E.g. PER GROUP(DEPT) OVER 4
* This then only matches the condition when the item itself matches a group total
* It returns the total number of products of ALL groups that match, but flags (using ?) only those items which match the basic condition AND the group condition.

Test Oper

* < : Less than to test var, returns 0 or 1
* <= : Less than or equal to test var, returns 0 or 1
* > : Greater than to test var, returns 0 or 1
* >= : Greater than or equal to test var, returns 0 or 1
* = : Equal to test var, returns 0 or 1
* <>: Not equal to test var, returns 0 or 1
* IN: In the list given by test var (can use lists and/or ranges). Returns 0 or 1.
* OUT: Opposite of IN, returns 1 or 0.
* EACH: Returns the test value or quantity divided by the test var rounded down.
* UPTO : Acts as ‘EACH 1’, but limits how many items are tagged (for limiting question answers).

The following macros are auto created on execution of a condition.

* {?} : The most recent condition, number of repetitions (e.g. 14 items with each 3) returns 4.
* {?n}: The value of {?} for condition n, starting from the first condition being {?1} in the behaviour.
* {@}: The most recent condition, number affected (e.g. 14 items with each 3) returns 12.
* {@n}: The value of {@} for condition n as per {?n}.
* {#n}: Returns the list of product ids and quantities matched by condition n. e.g. 3x3320+2x3319. This is ready for an add command if required.
* {@BATCH} the smallest value of {@} (items) in the active batch conditions
* {?BATCH} the smallest value of {?} (repetitions) in the active batch conditions.

# QUESTIONS

Questions in behaviours use the same system as questions asked from button commands, however they also cache the answer within the transaction so that a given question will only be asked once in a behaviour in a part.

The syntax of the Question command in behaviours is:

QUESTION [ONCE] [<name>] <macro>= <kind>(parameters…..)

This is equivalent to the following button command:

<macro>=Question(<kind,parameters)

The following parts exist in the behaviour question:

* Optional ONCE parameter, specifying use the same answer for all repetitions, without ONCE a new question will be asked for each repetition.
* <name> a unique name (unique across all behaviours) that determines if two questions are the same for the purposes of caching the answer. Once a detail has a value for the question of this name, and no matched details remain unanswered – then the question will not be asked again.
* <macro> a macro name to put the most recent return value into, NOTE that the most recent value may be blank if the question is cached.
* <kind> the question kind, with a few special cases. See the command reference manual for different question types (see below for a few special cases)
* Parameters – the parameters for the question command.

Special questions:

* CONTINUE – is just a yes/no do you wish to continue question.
* RANDOM – allows for generation of random values.
* SECONDARY – place a prompt on the secondary monitor if one is active.

Questions will store FALSE next to unanswered detail lines if no option is selected, or blank is returned or NO is answered on a continue. In each of these cases. Otherwise the returned value (or true) is stored for the question next to each unanswered detail line.

Any time there are no true answers to a continue question, the behaviour will stop operation at the question to which no true answers exist.

Any time an actual question responds with a blank or false, then no further questions will be asked (they will default to NO/false).

Special Macros created by questions (examples assume question name is ‘name’), these macros are for use in the quantity section and are all net absolute values of quantities in the current part, only counting the most recent condition matching.

* {?name} – refers to the total number of matched items with an answer to this question
* {?name@Y} – refers to the total number of yes answers to this continue question
* {?name@N} – refers to the total number of no answers to this continue question
* Note – see adjustments for how to use question answers
* Note – there should always be zero non answered items.

# ADJUSTMENTS

Adjustments are identical in behaviours as in button commands, however there are a few special macros and values that can be used. Note also, that whilst all adjustments can be made within a behaviour, it may be wise not to do some of them within behaviours.

Also note that as behaviours are executed, the adjustments are just collated, and prepared. They are not performed on the transaction until all behaviours are executed (special – see before and after timing above).

Macros can be used in adjustments, just as in any other command – however in addition there are special non-macro values that can be used as the parameters.

A normal adjustment command (except from Add) will be like this:

MODIFY [BATCH x/y] Price(<range>,<number>,<adjustment>

An optional batch section can be specified which ties in with the batch settings for the conditions, a batch modify will be skipped (and processing continues) if the batch id (x) exceeds the batch maximum (y).

The three parameters are:

* <range> this defines the range of products to use. In addition to this you can use ‘?’ or ‘?n’ to represent the products tagged by the last or the given condition (n). Note this is NOT a quantity of product, it is merely specifying the range of products to choose from. If this is ‘\*’ then all products are in the range.
* <number> this defines the quantity of product to change from within the range specified above. If this is ‘\*’ or ‘?’ then ALL products within the range will be adjusted.
* <adjustment> this defines the <relative>@<absolute> pricing formula to apply (e.g. -10%@COST).

Common adjustments within behaviours:

* Tax – creates a tax entry for the items using the given formula
* Loyalty – creates a loyalty entry for the items using the given formula.
* Price – adjusts the fluid pricing
* Base – adjusts the base pricing (be careful on this one).
* Instruction – adjusts the instructions on a given item.
* Add – add products to the transaction.
* Delete – delete products

Pricing formula consist of the following basic format:

[relative adjustment][@absolute amount]

The relative adjustment is “<s>nn.nn<f>” as follows, where nn.nn is a number (e.g. 1.44) and <f> is one of the following:

* $: specifies a dollar amount, this is the default if <f> is left out.
* %: specifies a percentage change (based on the specified absolute portion).

And <s> is one of the following, only when <f> is ‘%’:

* =: nn.nn% of the absolute amount (e.g. =20%@SELL means price is 20% of sell)
* ~: Reverse a percent of (e.g. ~10%@SELL, means the that the result is the value added to a value by adding 10% to end up with @SELL as the result). Used for calculating tax from the total.
* !:Determine the amount such that the percent GP is given
* +:Add nn.nn% to the value, this is the default (i.e. ‘+’ is optional). (e.g. 10%@COST means add 10% to the cost).

The absolute amount is one of the following:

* @Pricing/Level – e.g. @Default/1 – meaning use that price from the pricing list, or use the default one with the same Level (if the original is absent) if not use @SELL
* @SELL – the default sell price
* @COST – the stored cost price
* @FREE – zero
* @BASE – current base price
* @PRICE – current ‘price’ price (use with caution)
* @TAX – current tax (use with caution)
* @PROFIT – current profit on this item. (use with caution)

You can use smart macros in adjustments to account for questions answered over several iterations of the behaviour process by using the ‘ for Question[?ConditionId]` suffix on a modify line.

The question specifies a question that has been asked, and specifies that the modify be performed once for each detail in which the question stored a non false (i.e. either true or some value). The optional condition id specifies a condition id to filter the tested details (the default is to use the most recent question). The following examples illustrate how it can be used:

MODIFY Add({ProductId},1,@SELL)  
MODIFY Add({ProductId},1,@SELL) for \*   
MODIFY Add({ProductId},1,@SELL) for \*?2  
MODIFY Add({ProductId},1,@SELL) for Bonus6pkDeal   
MODIFY Add({ProductId},1,@SELL) for Bonus6pkDeal?2

# TODO

The todo command adds an item to the todo list for the transaction, these entries will be executed at the finalisation of the transaction. The syntax of the TODO command is:

TODO <ID> <NAME>|”<NAME WITH SPACES>” [MERGE] [<Quantity>] <Kind>(Parameters)

The following are the parts of the command:

* <ID> - a unique name across the transaction, this ensures that a given todo item is only added once, even if it occurs in different behaviours.
* <NAME> or “<NAME WITH SPACES>” – the name as it will appear on the receipt and transaction
* [MERGE] – optional flag to merge multiple of the same todo into one command at finalise (e.g. sum all the quantity and values for the same kind of todo)
* [<quantity>] – an optional quantity of this todo to create, defaults to 1.
* <Kind> - the kind of todo entry, currently only Issue commands are supported.
* Parameters – the parameters for the command

You can think of todo items as like commands but with tighter restrictions on what can be done.

# TEST VALUES (in Conditions)

The basic way conditions works leaves you with a left hand side an operator and a right hand side.

The left hand side with be a quantity, value or attribute you choose from the transaction or till such as “QUANTITY OF RECEIPT FOR GROUP(DEPT:BEER)”.

The operator might be EACH, > or < or = to name a few (see above).

The right hand side is the value you are comparing or testing against. You may use macros or parameters in this value.

There are special situations for when you use the IN operator for addition ways to specify the right hand side.

Depending on what the comparison is, the choices can vary. But fundamentally the comparison is as follows:

CONDITION <somedate> IN TestA,TestB,TestC … comma separated list

If any one of the tests is true, then the condition succeeds.

Each test can be of the form:

* Value (e.g. 47 or Beer)
* For numeric values - a range (e.g. 2-4)
* For date values – a space separated list of date component tests all of which must be true

Samples:

CONDITION <value> IN 100-200  
CONDITION <value> IN 1-9,99-100,500-1000  
CONDITION <value> IN BEER,WINE,SPIRITS

For date values the following components can be tested in the space separated list, not that in most test you can specify one value or a range (A-B).

* Specific day (dd/mm/yyyy or dd/mm/yy) specifies the specific date and year
* Date this year (dd/mm) specifies the date this year – for example 1/12
* Easter or date relative to Easter (Easter/Minus/n or Easter/Plus/n) is that many days away from Easter – for example Easter/Minus/2-Easter/Plus/1 specifies the Easter long weekend.
* Day of month (n/DOW/Month) specifying the nth DOW of the month – for example 1/Tues/Nov is Melbourne Cup day
* Month (Jan, November, Dec) – for example Jun-Aug is winter
* Day of Week (Fri, Saturday) – for example Monday-Friday is the working week
* 24 hour time – for example 1100z or 2200z represent 24 hour times
* 12 hour times – for example 8:30am, 2pm represent 12 hour times
* A year – 4 digits e.g. 2019 represents a year.
* Day of the month – e.g. 1st, 3rd, 14th, for example 7th-11th
* Past the hour – e.g. 8past-25past triggers for any time between 8 and 25 past the hour.

# ABORT

Abort is a special command that has the following syntax:

ABORT <message string>

If the behaviour executes this line, the behaviour processing and the error BehaviourAbort is raised with the message string provided.

This can be used to conditionally abort the adding of a product, as the adjustment function will revert all changes when this abort exception is raised.

Currently only the behaviour execution initiated by Adjustment commands will raise the abort exception, other executions will ignore the ABOIRT command in effect.