

# Guide - Understanding Macro Conditional Commands

---

The conditional commands are commands that allow you to create if/elseif/else/while.

## Introduction to conditional commands and expressions

The most basic forms of expressions in the macro system are stores. When working with stores think of them as expressions with a value. When you type "`@STORE1 = 5`", you're assigning '5' into `@STORE1`. '5', obviously, has the value 5, or in other words '5' is an expression with the value of 5.

After this assignment, you'd expect `@STORE1`'s value to be 5 as well, so if you wrote `@STORE2 = @STORE1`, you'd expect it to behave just as if you wrote `@STORE2 = 5`. In other words, `@STORE1` is an expression with the value of 5 as well. This is also how the macro system handles it.

A very common type of expressions are comparison expressions. These expressions evaluate to either false or true. Macro supports `>` (bigger than), `>=` (bigger than or equal to), `=` (equal), `!=` (not equal), `<` (smaller than) and `<=` (smaller than or equal to). These expressions are used inside conditional execution, such as if statements. The comparison expressions can be used on STORES, the dynamic syntax (`[$Item.Column.Type]`) and simple types like integers, strings and date.

## STORES

Stores are places where you can store data for later user.

You have ten stores available as default in the macro system and they can be used to store values in:

```
@STORE1 = 2;  
@STORE2 = 'My string';  
@STORE3 = SQL('SELECT CardCode FROM ...');  
@STORE1 = @STORE2;
```

The `@STORE` can be used in a macro command for example:

```
@STORE1 = 2;  
StatusBar(Store1 value is @STORE1|Warning);
```

Will show "Store1 value is 2" as a warning in the statusbar.

## IF

The if command allows for conditional execution of macro commands.

```
IF (expression)  
BEGIN
```

```
(statement)
END
```

As described in the section above about expressions, expression is evaluated to either true or false. If the expression evaluates to true, macro will execute the statement, and if it evaluates to false - it'll ignore it.

The following example would create a messagebox with “3 is bigger than 1” if 3 is bigger than 1:

```
IF (3 > 1)
BEGIN
    MessageBox(3 is bigger than 1);
END
```

Often you'd want to have more than one statement to be executed conditionally. You can group several commands into a group. For example, this code would display “3 is bigger than 1” if 3 is bigger than 1 and would then assign the value 1 to @STORE1

```
IF (3 > 1)
BEGIN
    MessageBox(3 is bigger than 1);
    @STORE1 = 1;
END
```

IF commands can be nested infinitely within other if statements, which provides you with complete flexibility for conditional execution of the various parts of the macro.

## ELSE

The else command can be used if you want to execute a statement if a certain condition is met, and a different statement if the condition is not met. This is what else is for. Else extends an if statement to execute a statement in case the expression in the if statement evaluates to false.

For example, the following example would display STORE1 is greater than STORE2 if @STORE1 is greater than @STORE2, and STORE1 is NOT greater than STORE2 otherwise:

```
IF(@STORE1 > @STORE2)
BEGIN
    MessageBox(STORE1 is greater than STORE2);
END
ELSE
BEGIN
    MessageBox(STORE1 is NOT greater than STORE2);
END
```

## ELSEIF

The elseif command is a combination of if and else. Like else, it extends an if statement to execute a different statement in case the original if expression evaluates to false. However, unlike else, it will execute that alternative expression only if the elseif conditional expression evaluates to true. For example, the following code would display STORE1 is bigger than STORE2, STORE1 equal to STORE2 or STORE1 is smaller than STORE2:

```
IF(@STORE1 > @STORE2)
BEGIN
    MessageBox(STORE1 is bigger than STORE2);
END
ELSEIF (@STORE1 = @STORE2)
BEGIN
    MessageBox(STORE1 is equal to STORE2);
END
ELSE
BEGIN
    MessageBox(STORE1 is smaller than STORE2);
END
```

## WHILE

The while command creates a loop in the macro. The basic form of a while command is:

```
WHILE(expression)
BEGIN
    (statement)
END
```

The meaning of a while statement is simple.

It tells Macro to execute the statement repeatedly, as long as the while expression evaluates to true.

The value of the expression is checked each time at the beginning of the loop, so even if this value changes during the execution of the statement, execution will not stop until the end of the iteration.

Sometimes, if the while expression evaluates to false from the very beginning, the statement won't even be run once.

```
@STORE1 = 1;
WHILE (@STORE1 <= 10)
BEGIN
    StatusBar(Loop executing @STORE1|Warning);
    @STORE1 = @STORE1 + 1;
END
StatusBar(Loop done|Warning);
```