Table of Contents

CREATE 2
Syntax 2
Example 2

Actions, Action List, Business Object, Process, References, Lists

CREATE

This action creates an instance of the specified business object.

Syntax

CREATE Id() or CREATE ArithmeticOperation() [NO VALIDATION][NO RULES]

where Id() is identifier of the business object to create, for example

```
CREATE Account
```

and Arithmetic Operation is an expression producing a string indicating the name of the object to create.

Note that the identifier here is not the attribute identifier and thus may not indicate a reference. The action is also not applicable to business object groups.

CREATE action can optionally initialize one or more attributes of the business object that it creates.

Example

```
CREATE Account WITH Account.State='Open', Account.Balance=0
```

The attributes to be initialized must follow the WITH keyword and must be separated by comma. Any arithmetic operation is valid as the initialization expression.

Sometimes it may be necessary to create several instances of a business object with a single CREATE action. The following constructs are supported:

- CREATE Id() FOR EACH AttributeIdentifier()

This action will create as many instances of the object as there are instances of another business object in the Context. For example,

```
CREATE Transaction FOR EACH Account
```

- CREATE Id() FOR EACH (DAY | WEEK | MONTH | QUARTER | YEAR | WEEK_DAY | WEEKEND_DAY) BETWEEN ArithmeticOperation() AND ArithmeticOperation

This action will create as many instances of the specified business object as there are days(weeks/months/quarters/years) in the specified date interval. Note that the Arithmetic Operations here must be operations on dates, for example,

```
 CREATE Transaction FOR EACH DAY BETWEEN Account.OpeningDate AND
Account.ClosingDate
```

The date interval includes both starting and ending dates.

Note also that if the object being created has any attributes of the Date type, these attributes can be initialized with the value of the date for which the object is being created. For example,

```
 CREATE Transaction FOR EACH DAY BETWEEN Account.OpeningDate AND
Account.ClosingDate WITH Transaction.AppliedDate=Day
```

Every transaction created in this way will have the AppliedDate attribute initialized to the day for which the transaction is created.

- CREATE Id() FOR EACH NUMBER BETWEEN ArithmeticOperation() AND ArithmeticOperation

This action will create as many instances of the specified business object as there are numbers in the specified interval. Note that the Arithmetic Operations here must be operations on numbers, for example,

```
 CREATE Transaction FOR EACH NUMBER BETWEEN 1 AND 3
```

The interval includes both starting and ending numbers.

- CREATE Id() FOR EACH FILE IN ArithmeticOperation()

This action will create as many instances of the specified business object as there are files in the specified directory. The Arithmetic Operation used in the expression must produce a string specifying the directory in which the system will look for files. For example,

```
 CREATE Attachment FOR EACH FILE IN 'c:/mydirectory'
```

Note also that if the object being created has any attributes of the Document type, these attributes can be initialized with the file for which the object is being created. For example,

```
 CREATE Attachment FOR EACH FILE IN 'c:/mydirectory' WITH
Attachment.Document=File
```

Every attachment object created in this way will have the Document attribute initialized to the file for which the attachment is being created.

- Variation of the above expression is: CREATE Id() FOR EACH NEW FILE IN ArithmeticOperation()

The only difference with the previous expression is that the file in the specified directory is deleted after the business object has been created.

The NO VALIDATION keyword creates an object without executing validation rules (rules that REPORT ERROR). This can be useful if you are creating just a template of the object providing all required values later.

The NO RULES keyword creates an object without executing any rules at all. Be careful with this

keyword as this may create an instance of the object with invalid data.