

# Magic Software AppBuilder

Version 3.2

## Scripting Tools Guide

Corporate Headquarters:

Magic Software Enterprises  
5 Haplada Street,  
Or Yehuda 60218, Israel  
Tel +972 3 5389213  
Fax +972 3 5389333

© 1992-2013 AppBuilder Solutions

All rights reserved.

Printed in the United States of America.

AppBuilder is a trademark of AppBuilder Solutions. All other product and company names mentioned herein are for identification purposes only and are the property of, and may be trademarks of, their respective owners.

Portions of this product may be covered by U.S. Patent Numbers 5,295,222 and 5,495,610 and various other non-U.S. patents.

The software supplied with this document is the property of AppBuilder Solutions and is furnished under a license agreement. Neither the software nor this document may be copied or transferred by any means, electronic or mechanical, except as provided in the licensing agreement.

AppBuilder Solutions has made every effort to ensure that the information contained in this document is accurate; however, there are no representations or warranties regarding this information, including warranties of merchantability or fitness for a particular purpose. AppBuilder Solutions assumes no responsibility for errors or omissions that may occur in this document. The information in this document is subject to change without prior notice and does not represent a commitment by AppBuilder Solutions or its representatives.

1. Scripting Tools Guide .....	2
1.1 Introduction to Scripting Tools .....	2
1.2 TurboScripter Object's Model Reference .....	2
1.2.1 Starting and Using TurboScripter .....	2
1.2.2 Tracing Information about a TurboScripter Session .....	3
1.2.3 TurboScripter Object Reference .....	6
1.2.4 Valid Domain Types and Values .....	16
1.3 Starting TurboCycler .....	20
1.4 Using Default Templates .....	23
1.5 TurboCycler Tutorial .....	30
1.6 Using TurboCycler Developer Kit .....	39
1.6.1 Creating a Generation Template .....	39
1.6.2 Editing and Compiling a Template .....	41
1.7 TurboCycler Template Language .....	47
1.7.1 Flow Diagrams Overview .....	48
1.7.2 Template Language Statements .....	49
1.7.3 Other Blocks .....	59
1.7.4 Template Block Features .....	63
1.7.5 Other Statements .....	64
1.7.6 Supporting Statements and Expressions .....	68
1.7.7 Functions for TurboCycler Developer's Kit .....	85
1.7.8 Template Samples .....	89
1.8 TurboCycler Repository Types and Properties .....	96
1.8.1 Repository Object Type Query Sample .....	96
1.8.2 Object Types and Properties .....	97
1.8.3 Entity Types and Properties .....	98
1.8.4 Entities and Relationships .....	110
1.8.5 Relationship Types and Properties .....	140
1.9 TurboCycler Window Controls .....	150
1.9.1 Creating Controls .....	150
1.9.2 Property Types and Property Values .....	151
1.9.3 Window Control Properties .....	154
1.9.4 Window Properties Matrix .....	161
1.9.5 Window Statement Control Types .....	163
1.10 Build Scripts .....	171

# Scripting Tools Guide

AppBuilder includes two tools for scripting for increased productivity and a means to create and enforce corporate standards in development environments. The AppBuilder installation comes with sample scripts for both tools.

This guide describes the AppBuilder Scripting tools and includes the following topics and sections:

- [Introduction to Scripting Tools](#)
- [TurboScripter Object's Model Reference](#)
- [Starting TurboCycler](#)
- [Using Default Templates](#)
- [TurboCycler Tutorial](#)
- [Using TurboCycler Developer Kit](#)
- [TurboCycler Template Language](#)
- [TurboCycler Repository Types and Properties](#)
- [TurboCycler Window Controls](#)
- [Build Scripts](#)

## Introduction to Scripting Tools

AppBuilder includes two tools for scripting for increased productivity and a means to create and enforce corporate standards in development environments. The AppBuilder installation comes with sample scripts for both tools.

TurboScripter is non-proprietary software based on an industry standard that provides a repository interface that can be accessed from the AppBuilder Construction Workbench or from third-party tools that use the Component Object Model (COM). TurboScripter can be used from or integrated with external tools, including Microsoft Office. TurboScripter uses VBScript or JScript to access the properties and methods in AppBuilder.

TurboCycler is coupled with the AppBuilder repository-based software and uses a proprietary scripting language to manipulate the AppBuilder repository objects using less lines of code. Its source compiles to a binary file with a very fast compiler. TurboCycler can provide an easy way to create object hierarchies in the repository. TurboCycler also comes with a number of templates and a Developer's Kit.

## TurboScripter Object's Model Reference

TurboScripter is a non-proprietary software program that is based on industry standard. It is an optional tool that provides a mechanism by which scripts written in VBScript or JavaScript can be used to access the repository, manipulate repository objects, and access other applications, such as Microsoft Word, that function through automation. TurboScripter provides a scriptable repository interface that can be accessed from the AppBuilder Construction Workbench or from other tools through the Component Object Model (COM). Tools within the Construction Workbench have menu options to invoke TurboScripter and pass repository objects to it.

Details of the TurboScripter object model are covered in this section under [TurboScripter Object Reference](#).

## Starting and Using TurboScripter

You can use TurboScripter to automate many of the repetitive tasks in the software development lifecycle. TurboScripter uses VBScript or JScript to access the properties and methods in AppBuilder.

Auto.vbs is an example of a script that counts all the data in the repository. It is written and stored in a directory that TurboScripter can access. To start auto.vbs, complete the following steps:

1. Select *Start > All Programs > AppBuilder > Construction Workbench* .

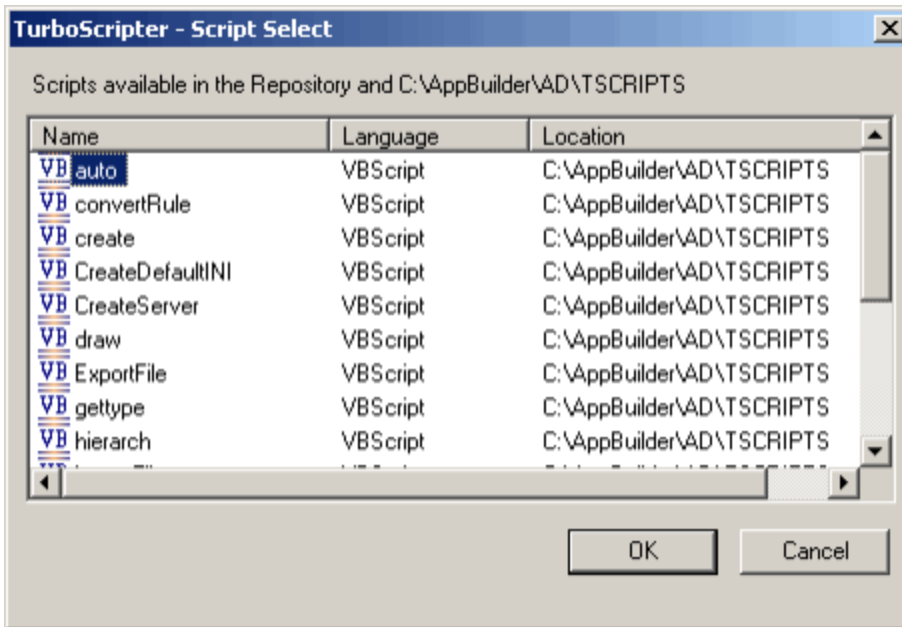
The Construction Workbench window appears.

2. Type your username and password in the User Name and Password fields and click *Connect* .

3. Click *Tools > TurboScripter* .

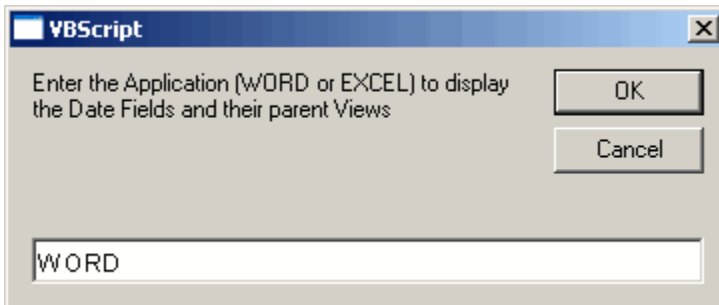
The Script Select window appears.

***TurboScripter - Script Select Window***



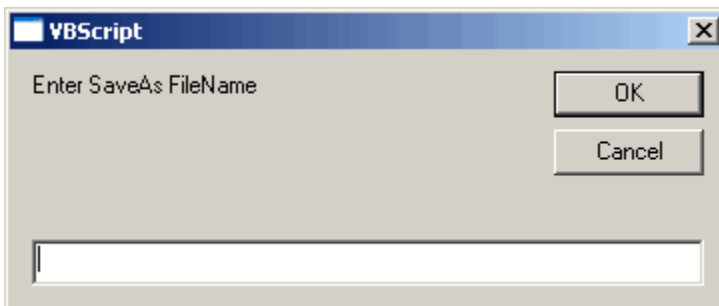
4. In the TurboScripter - Script Select window, select the script file that you want to run and click *OK* .
5. An information window appears telling you the count of all date fields in the repository. Click *OK* to close the message window.
6. In the VBScript dialog, type the application-Word or Excel-to display the date fields and their parent views and click *OK* .

**TurboScripter VBScript window**



7. An information window appears. Click *OK* to close the message window.
8. In the VBScript dialog, type the filename to save the file and click *OK*.

**TurboScripter VBScript file save window.**



## Tracing Information about a TurboScripter Session

The Trace file name and path are determined by the Registry settings for Trace\_File under the TurboScripter Key. You must select a hierarchy object to make the TurboScript menu item available. The default trace file is *ts.out* in the AppBuilder\AD\SCRIPTS directory. The amount of

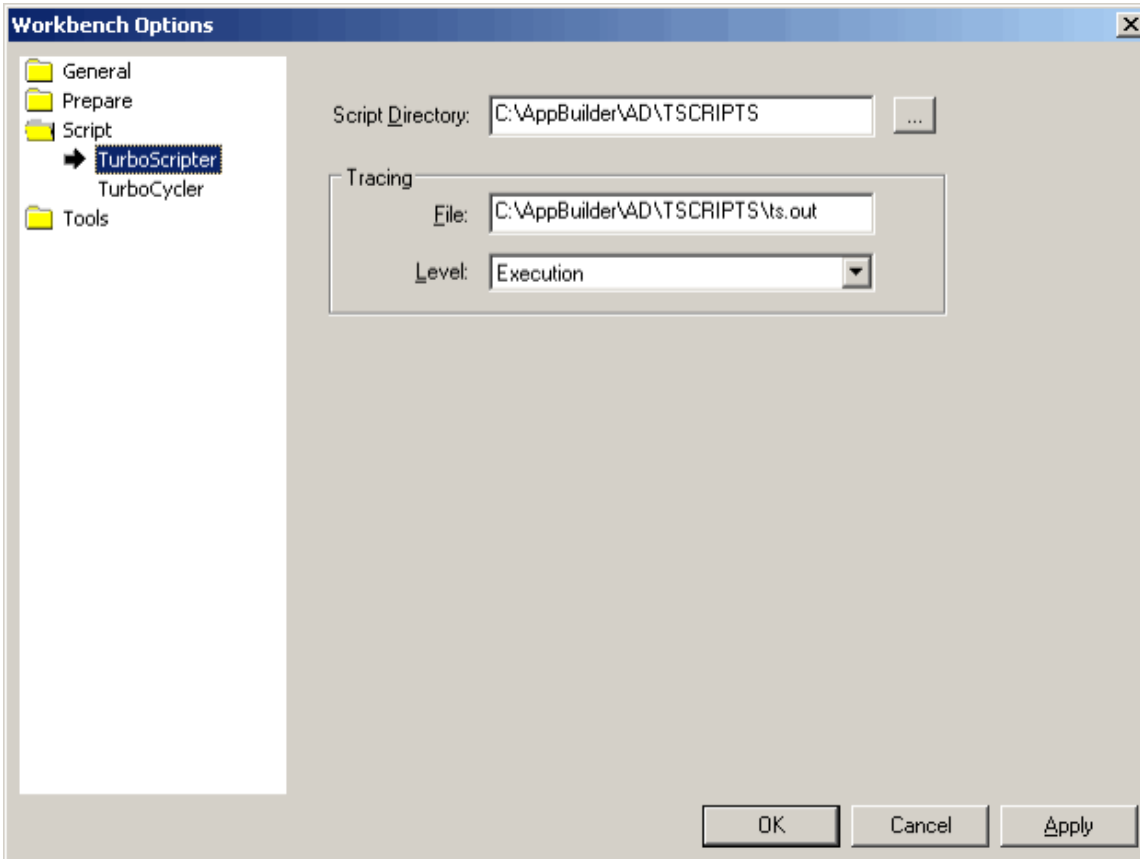
information in this file is dependent on the TRACE LEVEL settings in the TurboScripter Options dialog. If you are debugging a script, you can set the Trace level to DEBUG; however, performance might be affected.

### To trace information about a TurboScripter session

1. Click **Tools > Workbench Options > Script > TurboScripter**.

The TurboScripter set of options appears. See [TurboScripter VBScript window](#).

#### *The TurboScripter set of options in the Workbench Options window*



2. Type the Script Directory or click the button to the right of the field to browse for the correct directory.

3. Type the directory where you want to save the tracing file.

4. Select the tracing level and click *OK*.

The following trace levels are available:

- None = no tracing
- Execution = traces errors only
- Debug = traces warnings, errors, and function
- All = traces what the program is doing

### Accessing the Repository Using VBScript or JavaScript

Scripts in TurboScripter access the repository through the Repo object. Repository objects can be queried (for example, Repo.Query("FIELD", "RETURN\_CODE") or created (for example, Repo.Create("RULE", "TS\_SAMPLE") from a Repo object. These repository objects are returned as TSEnt objects for entities and TSRel objects for Relations.



The Repository Service Manager module, gresvcnt.exe, must be running before you can connect to a Personal Repository or a Workgroup Repository using TurboScripter.

### Accessing the Properties of an Entity or Relation Object

Use *GetProperty* or *SetProperty* to get and set properties of an object. For example, you could use *myObj.GetProperty( Name )* or *myRel.SetProperty( Sequence\_Number )*. The property names used in these methods must be valid property names according to the Workgroup Repository model. You can get a list of entities, relations, and their property names by running the sample script *Model.vbs* that comes with TurboScripter, or you can use the *Object.Property* notation to access the properties of an object. For example,

*objName = myObj.Name , myRel.Sequence\_Number = n .*



The object and property names might differ from those in this documentation. TurboScripter uses the exact names defined by the Workgroup Repository.

## Invoking Other Scripts from a Currently Executing Script

Call *Host.CreateScript(ScriptFileName, Language)* creates a new script object and initializes it to the current environment. Once this is done, you can pass repository objects to the script by calling *AddInputObject* on the new script object. The new script is run by calling the *Run* method.

## Manipulating or Creating Windows and Controls within the Windows

Use the *CoPanel* Object to create, update, read, and delete controls within a window. The *CoPanel* Object exposes the controls and attributes of the window stored in a panel file. The sample scripts *pnl\_read.vbs*, *pnl\_crt.vbs* demonstrate access panel objects/files.

## Accessing Drawing Files in the Repository

VBScripts or JScripts can read the repository objects stored within a drawing file. This is done through the *CoDraw* object. The sample script *draw.vbs* demonstrates how to access an Entity Relationship Diagram (ERD) and retrieve the repository objects within the drawing. The drawing file is retrieved from the repository and might not reflect uncommitted changes made to that file in the workbench.

## Accessing the Repository from VB

You can access the repository from VB or any other tool that supports COM. You must create the *TurboHost* object either through the *CreateObject* method or by declaring a variable of type *TurboHost*. To declare a variable you first must add the *TSATL* library to the references for the VB project. When the *TurboHost* object has been successfully created, call *Repolnit* on this object to establish a repository session. This method returns the *Repo* object on success. Now, you can create or query objects from the repository and manipulate just as you would from a script. The sample project *VB\_TSRepo* that comes with TurboScripter in the same directory as the *TurboScript* sample scripts demonstrates this.

## Accessing Repository Objects Passed from the Construction Workbench

You can access the "InputObject" Dictionary object that contains the list of repository objects passed by the Construction Workbench. This Dictionary has a 0 based index as its key and *TSEnt* objects as the items.

## The Dictionary Object

The Dictionary Object is a container type created by Microsoft that contains a list of Key Item pairs. In most cases, TurboScripter uses the Key as a zero-based index, and the item contains the actual value. Further information about the Dictionary object is available at the Microsoft Web site or in VBScript documentation.

Example:

**REM \*\* Create a Dictionary object to set the properties of an object on creation**

```
Set myProps = CreateObject("Scripting.Dictionary")
REM ** The property name is the KEY and the value is the ITEM
MyProps.Add("Field_Type", "Small Integer or Integer")
MyProps.Add("Fld_Len", "15")
Set myObj = Repo.Create("FIELD", "TS_TEST", MyProps)
```

## Obtaining a Relation Object Between Two Objects

When calling *GetChildren* or *GetParents*, pass in an empty Dictionary Object as the optional third parameter. When the call returns, this object contains the relations corresponding to the children for the same Key value. (The Relation for the first Child object in the returned Children object is the first element in the Relations object. Relation for the second child is the second element in the Relations object, and so on.)

## Creating or Modifying Files under a Component\_Folder

If the *Component\_Folder* does not exist, complete the following steps:

1. Create a *Component\_Folder* object using *Repo.Create*.
2. Create a child of type "WEAK\_ENTITY" through the relation "COMPONENT\_FOLDER\_CONTENT".

3. Set the PathName property of the COMPONENT\_FOLDER\_CONTENT relation to the full path name of the file you want to store in the repository.
4. Call SetFile method on the Relation object passing in the text string containing the file contents and the type, which is 17 (ComponentFolderFile).



VBScript and JScript versions 5.0 and later do not support binary files. This restricts the use of TurboScripter to text files.

## TurboScripter Object Reference

This section contains brief descriptions of the TurboScripter objects for reference. Each entry has a description of what the object does and lists the filename of a sample script utilizing that object that can be found in the ADTSCRIPTS subdirectory. [vb sample scripts provided with TurboScripter](#) lists the TurboScripter sample scripts with a description of each script.

### *vb sample scripts provided with TurboScripter*

Sample VB script	Description
auto	Retrieves fields that have a date type and documents the views that use them in an Excel spreadsheet.
create	Creates or deletes repository objects.
CreateDefaultINI	Creates new appcofgdefault.ini and partitiondefault.ini files from downloaded AppBuilder mainframe files.
CreateServer	Creates a Server object. Searches all Rules in the repository and adds frontier rules to the Server object. If more than 999 rules are found, it creates additional Server objects.
draw	Opens a specified ERD file from the repository and displays the names of entities in the diagram.
ExportFile	Retrieves a file from an object and saves it. The sample can retrieve Rule source file, Window panel file, and an .ini configuration file from AppConfig and Partition.
gettype	Gets all the Views of a certain Rule and outputs their relationships and usage to the parent Rule.
hierarch	Generates a sample hierarchy beginning with TS_. In the process, it uses Create, CreateChild, SetProperty, SetFile, Property Dictionaries during Create and for Relations during CreateChild, Commit, and Rollback.
ImportFile	Restores file to object. The sample can set rule source file, window panel file, and an .ini configuration file for AppConfig and Partition.
model	Retrieves the meta-information about the repository.
pnl_crt	Creates a sample panel in the repository and creates a few sample controls with properties.
pnl_read	Retrieves a Window Panel that you want to read from the repository, opens it, and displays the controls and their properties.
query	Prompts you for the method that you want to test, and, depending on the method entered, runs Query, Find, or Get method.
rename	Renames a view or a field and modifies any rule source that uses the original name.
SortSet	Sorts the selected set by the values in the Define column of the set. This script must be invoked from the AppBuilder Workbench, with the set to be sorted selected.

The TurboScripter objects include the following:

- [Repository Object - Implements IRepo](#)
- [Entity Object - Implements ITSEnt](#)
- [Relation Object - Implements ITSRel](#)
- [Trace Object - Implements Itrace](#)
- [Host Object - Implements IScriptUtils](#)
- [Child Script Object - Implements IScript](#)
- [TurboHost - Implements ITurboHost](#)
- [CoDraw Object](#)
- [CoPanel Object](#)
- [Trace Object - Implements Itrace](#)
- [CoCtrl Object](#)
- [Attribute Object](#)



## Repository Object - Implements IRepo

This section discusses the Repository Object - Implements IRepo. The following sections describe actions that can be taken on the Repository Object and properties that can be applied to the Repository Objects.

### **Query(String ObjectType, String PropName, String Pattern)**

Queries the repository for Objects of type ObjectType with the property PropName that have the value Pattern. Returns a Dictionary object containing a list of TSEnt objects that the repository returns. This list is keyed by the index of the elements in the list starting from 0. Pattern can contain '\*' as wildcard. If the Property name is a Domained value, use the Display Value in the repository model.

Example:

```
REM *** Following query returns a Dictionary of TSEnt objects of type
REM *** "VIEW" that have names beginning with "HPS"
Set myViews = Repo.Query("VIEW", "Name", "HPS")
REM *** Following query returns a Dictionary of TSEnt objects of type
REM *** "FIELD" that have their "Field_Type" property set to "8"
REM *** "Field_Type" is a Domain and "Date" is the Display Value
Set myDateFields = Repo.Query("FIELD", "Field_Type", "Date")
```

Sample script: Query.vbs

### **Find(String ObjectType, String PropName, String Value)**

Returns the corresponding TSEnt Object. If more than one object matches the criteria, the first object is returned. If the Property name is a Domained value, use the Display Value in the repository model. This is typically used to find a specific object whose unique property value is known.

Sample script: Query.vbs

### **Get(String Type, String Name)**

Returns the TSEnt object with the given Type and Name.

Sample script: Query.vbs

### **Create(String Type, String Name, (optional)Dictionary Properties)**

Returns the created object. If the TSEnt object already exists, the existing object is retrieved and returned. Display values can be used for Domained properties. To set the properties when creating an object, fill in the Properties object with the Key as Property Name and Item as Property Value.

Sample script: Create.vbs

### **Delete(String Type, String Name)**

Returns Nothing. Raises Error if the specified object is not found.

Sample script: Create.vbs

### **Commit()**

The entire session is committed. When called from the WorkBench, changes done by the other tools are also committed.

Sample script: Create.vbs



Attempting to access deleted objects after commit could destabilize the Construction Workbench. Make sure that objects that have been deleted before are not accessed after the commit.

### **Rollback()**

The entire session is rolled back. When called from the Construction Workbench, changes done by the other tools are also rolled back.

Sample script: Create.vbs



Attempting to access objects that have been created before the rollback could destabilize the Construction Workbench. The script author must make sure that objects that have been created before are not accessed after the rollback.

### **GetEntityTypes()**

Returns a Dictionary object containing all the entity types in the repository as strings. This list is keyed by the index starting from 0.

Sample script: Model.vbs

#### ***GetRelationTypes()***

Returns a Dictionary object containing all the relation types in the repository as strings. This list is keyed by the index of the elements in the list starting from 0.

Sample script: Model.vbs

#### ***GetPropertyNames(String ObjectType)***

Returns a Dictionary object containing all the property names for this object type. This list is keyed by the index of the elements in the list starting from 0.

Sample script: Model.vbs

#### ***GetPropertyType(String ObjectType, String PropName)***

Returns the property type as a String. The value is one of "STRING", "LONG", "UNSIGNED\_INT", "DOMAIN" or "INTERNAL"

Sample script: Model.vbs

#### ***GetDomainValues(String ObjectType, String PropName)***

Returns a Dictionary object containing all the valid Domain values for this property. This Dictionary Object contains the Storage Values of the domain elements as the Key and the Display Value (or User Value) as the item. Returns an error for properties that do not have a Domain associated.

Sample script: Model.vbs

#### ***StatusToString(int status)***

Returns the repository status message from a status code.

### **Entity Object - Implements ITSEnt**

#### ***GetProperty(String PropName)***

Returns the required property as a Variant object. The only variable type in VBScript. The PropName must be a valid property of this repository object type as published in the repository model.

Sample script: Query.vbs

#### ***SetProperty(String PropName, Variant PropValue)***

Sets the property specified by PropName to the value in PropValue. Returns nothing. Raises an Error if the property is invalid or the value is invalid. For properties that have a domain associated with them, the Display value (or User Value), not the storage value, must be specified.

Sample script: Hierarch.vbs

#### ***CreateChild(String Type, String Name, String RelationType, (optional) Dictionary Properties, (optional) Dictionary RelationProperties)***

Creates a child object of the specified Type and Name. The Properties Dictionary can contain the properties of the child object while the RelationProperties Dictionary can contain the properties of the relation, for example, Sequence Number of the relation. If the specified object already exists in the repository, this method returns the retrieved object.

Sample script: Hierarch.vbs

#### ***GetChild(String Type, String Name, String RelationType)***

Returns an Entity Object (TSEnt) that is a child of the calling object.

#### ***GetChildren(String Type, String RelationType, (optional) Dictionary RelationObjects)***

Returns a Dictionary object containing the child objects (TSEnt) of the specified type. The indices for these objects are from 0 to (Count - 1). If the RelationObjects is a valid Dictionary, then the Relation object (TSRel) for each of the corresponding child objects are filled in.

Sample script: gettype.vbs

#### ***GetParents(String Type, String RelationType, (optional) Dictionary RelationObjects)***

Returns a Dictionary object containing the parent objects(TSEnt) of the specified type. The indices for these objects are from 0 to (Count - 1). If the RelationObjects is a valid Dictionary, then the Relation object (TSRel) for each of the corresponding child objects will be filled into RelationObjects. Use the same key value to access the child object as well as the relation object for that child.

Sample script: Rename.vbs

#### ***GetType()***

Returns the Object Type name of this object as a String.  
Sample script: gettype.vbs

***GetFile(Integer FileType)***

Returns the File associated with this object. Valid values for File Type are given in the following table:

***File Type Values***

Value	File Type
0	RepoKeywords
1	RepoText
2	WindowPanel
3	ReportSection
4	PhysicalBitmap
5	DrawingFile
6	ReportFile
7	RuleSource
8	ComponentSource
9	WindowHelp
10	DatabaseOptions
11	ServerOptions
12	MachineOptions
13	FormContents
14	MigrationFile
15	ClosureScope
16	ApplicationFolder
17	ComponentFolderFile
18	ConstructorSource
19	DestructorSource
20	MethodSource
21	CursorSqlBody
22	RepoInitializer
23	PrepFile

Example:

If this is a Rule object, then GetFile(7) will return the source file for this rule as a string.0(RepoText) is a valid file type for all Entity (TSEnt) and Relation (TSRel) objects.1(RepoKeywords) is a valid file type for all TSEnt objects.

Sample script: Rename.vbs

***SetFile(Integer FileType, String FileStr)***

Sets the file associated with this object in the repository to FileStr. This can be used to create RuleSource (Value = 7) files from RULE objects and RepoText(0) and RepoKeywords(1) files from all Entity objects (TSEnt).

Sample script: Rename.vbs

***DeleteChildRelation(String ChildType, String ChildName, (optional) String RelType)***

Deletes the Relation between this Entity object (TSEnt) and the child object identified by ChildType and ChildName. If this object and the Child object are connected by more than one type of relation, then the RelType must be specified.

**DeleteParentRelation(String ParentType, String ParentName, (optional) String RelType)**

Deletes the Relation between this Entity object (TSEnt) and the parent object identified by ParentType and ParentName. If this object and the parent object are connected by more than one type of relation, then the RelType must be specified.

**Properties implemented by TSEnt**

TSEnt does not have any static properties of its own. It assumes all the valid properties of the repository object that it represents.  
Example:

**Get the Field\_Type property of myField**

```
Set myField = Repo.Get("FIELD", "RETURN_CODE")
MyFieldType = myField.Field_Type
```

**ExtractODF(String directory)**

Extracts the Object Definition File (definition of the object in XML format) for this TSEnt into the given directory. The odf file name is in the format <ShortName>.odf.xml.

Example:

```
myObj.ExtractODF("C:\temp")
```

**ODFGen(String extractToDir, BOOL isDeep)**

Extracts the ODF for this TSEnt and its children in the same file when isDeep is True.

Example:

```
myObj.ExtractODF("C:\temp", true)
```

**IsPrepared**

Returns true if this TSEnt is already prepared.

**IsSystem**

Returns true if this TSEnt is a system object.

**IsTransformed**

Returns true if this TSEnt is a transformed object.

**MarkPrepared**

Marks this TSEnt as prepared.

**MarkDirty**

Marks this TSEnt as dirty.

**Relation Object - Implements ITSRel**

**GetProperty(String PropName)**

Returns the required property as a Variant object. The only variable type in VBScript. The PropName must be a valid property of this repository object type as published in the repository model.

Sample script: gettype.vbs

**SetProperty(String PropName, Variant PropValue)**

Sets the property specified by PropName to the value in propValue. Returns nothing. Raises an Error if the property is invalid or the value is invalid. For properties that have a domain associated with them, the Display value (or User Value), not the storage value, must be specified.

Sample script: hierarch.vbs

**GetType()**

Returns the Object Type name of this object as a String.

Sample : pnl\_read.vbs

### ***GetFile(Integer FileType)***

Returns the File associated with this object.

Example: If this is a COMPONENT\_FOLDER\_CONTENT object, then GetFile(17) returns the component folder file for this relation object as a string if it is a text file.

Sample script: rename.vbs

### ***SetFile(Integer FileType, String FileStr)***

Sets the file associated with this object in the repository to FileStr. This can be used to create ComponentFolderFile files from COMPONENT\_FOLDER\_CONTENT relation. When retrieving Component Folder files, only text files can be retrieved. Binary files cannot be retrieved.

Sample : rename.vbs

## **Host Object - Implements IScriptUtils**

### ***CreateScript(String FileName, String Language)***

Returns a Script object that is initialized to have the same environment as the calling script. This call is used to create a new script object that can be executed from inside this script. Call the method Run() on the returned Script Object to execute the new child script.

## **Child Script Object - Implements IScript**

### ***Run()***

Executes the Child Script object in the same environment as the parent script. This method raises an Error if the execution of the script fails.

### ***AddInputObject(String Type, String Name)***

Adds the repository object specified by Type and Name to the InputObject Dictionary of this child script object.

### ***AddTypeLib(String TypelibID, Long MajorVersion, Long MinorVersion, (optional) Long Flags)***

Adds the specified TypeLibrary to the scope of this script object. Any constants defined in the TypeLibrary can now be used within the script.

### ***AddTopLevelObject(String Name, Object TopObject)***

Adds the specified ActiveX Object to the list of Top Level objects for this script. The script can access the object using the Name parameter.

## **TurboHost - Implements ITurboHost**

The TurboHost object implements the functionality required to host the scripting engine and serves as the entry point to the Repo, Script and Trace Objects. The methods and properties of this object can be used from other applications like VB that support COM and are not for use from inside a script.

### ***Repolnit(String RepositoryName, String UserName, String Password, (optional) String VersionName)***

Connects to the specified repository and returns the repository object. This object is a must either to run scripts or work with any repository objects directly.

### ***ScriptRun(String ScriptFileName, String Language, Dictionary InputObjects, Long WindowHandle, (optional) Dictionary TopLevelObjects)***

Executes the specified script. Returns an error on failure. VBScript and JScript are the supported languages. The Window Handle can be 0, in which case the handle of the currently active window is used. The TopLevelObjects Dictionary contains ActiveX objects keyed by their name. These objects can be used from inside a script just like the default Top Level Objects like "Trace" or "Repo".

### ***ScriptletRun(String ScriptContents, String Language, Dictionary InputObjects, Long WindowHandle, (optional) Dictionary TopLevelObjects)***

Works the same way as ScriptRun except that a string containing the script instead of the file name is passed. This is useful when scripts are generated on the fly by an application.

### ***ScriptSelectAndRun(Dictionary InputObjects, Long WindowHandle)***

This method brings up a dialog box containing the scripts in the directory specified in the registry. This dialog also shows the scripts present in the TScripts Component Folder. You can then select the script you want to be executed.

### ***Scriptlnit(String ScriptFileName, String Language)***

This method returns the Script Object that is initialized to the file name and language specified. Once this object is obtained, methods to add input objects, to add top-level objects, and to execute the script can be called.

Top-level objects added by TurboScripter to the scope of all scripts:

- [Repository Object - Implements IRepo](#) - Repository Object for querying, creating or deleting objects in the repository
- [Trace Object - Implements Itrace](#) - Trace Object to get and set the trace file and trace level
- [Host Object - Implements IScriptUtils](#) - Host Object to enable creating a child script
- Input or [Entity Object - Implements ITSEnt](#) - Dictionary object that contains the input Entity objects (TSEnt). Sample script:Rename.vbs.

## CoDraw Object

The CoDraw Object provides access to Drawing Objects stored in the repository. Only read access is allowed. To access Drawing Objects from VB or TurboScripts, call CreateObject("CoDrawFile.CoDrawFile"). This returns the created ICoDrawfile object, which can be used to perform read operations on the drawing.

### **Open(String DrawingName, String DrawingType)**

Retrieves the specified Drawing object and opens it for reading. The DrawingName is the name of the Drawing object. The DrawingType is the value in the Drawing\_Type property of this object. Since the Drawing\_Type property can only have values from the Drawing\_Type domain, the Display value must be specified.

Sample script: Draw.vbs

### **GetAllRepoObjects()**

Retrieves all the repository objects in the drawing and returns a Dictionary object that contains these as Entity objects (TSEnt).

Sample script: Draw.vbs

### **GetRepoObjects(String Type)**

Retrieves the Repository objects of the specified Type in the drawing and returns a Dictionary object that contains these as TSEnt objects.

Sample script: Draw.vbs

### **DrawTrace**

Sets the trace level for the CoDraw object. The domain values are the same as the [Trace Levels](#) in the TurboScripter Object Model. The default value is set to TRACE\_NONE. You must set tracing explicitly to get trace information.



The CoDraw object reflects the state of the Drawing Object as it is in the repository. If changes are made to the drawing file from the workbench, these changes must be committed before the CoDraw Object is opened.

## CoPanel Object

The CoPanel object provides COM/OLE Automation-based access to the various controls and their properties. It enables a TurboScript to create, read, update, and delete the controls and their properties. They include:

- [CoPanel Object Control Types](#)
- [CoPanel Object Colors](#)
- [CoPanel Object GUI Types](#)



The constants in the following table can only be used when the script is invoked from within the Construction Workbench.

### **CoPanel Object Control Types**

	<b>Control Type</b>	<b>Constant</b>
1.	CHARTWINDOW	ctrlChart
2.	CHECKBOX	ctrlCheckbox
3.	COMBOBOX	ctrlCombobox
4.	EDITFIELD	ctrlEditfield
5.	ELLIPSE	ctrlEllipse

6.	GROUPBOX	ctrlGroupbox
7.	HOTSPOT	ctrlHotspot
8.	LISTBOX	ctrlListbox
9.	MULTILINEEDIT	ctrlMultiLineEdit
10.	OLECONTROL	ctrlActiveX
11.	PUSHBUTTON	ctrlPushbutton
12.	RADIOBUTTON	ctrlRadiobutton
13.	RECTANGLE	ctrlRectangle
14.	SPREADSHEET	ctrlMCLB
15.	STATICTEXT	ctrlStatictext
16.	BITMAP	ctrlBitmap
17.	MENU	ctrlMenu
18.	SUBMENU	ctrlSubMenu
19.	MENUITEM	ctrlMenuItem
20.	SEPARATOR	ctrlMenuSeparator
21.	CELL	ctrlCell
22.	CELLFIELD	ctrlCellField
23.	CELLTEXT	ctrlCellText
24.	XDATA	ctrlXData
25.	YDATA	ctrlYData
26.	FOOTINGTEXT	ctrlFootingText
27.	LEFTMARGINTEXT	ctrlLeftMarginText
28.	RIGHTMARGINTEXT	ctrlRightMarginText
29.	HEADINGTEXT	ctrlHeadingText

**CoPanel Object Colors**

	<b>Color</b>
1.	Default
2.	Black
3.	White
4.	DarkGray
5.	Gray
6.	LightGray
7.	DarkBlue
9.	DarkGreen
10.	Green
11.	DarkCyan
12.	Cyan
13.	DarkRed

14.	Red
15.	DarkMagenta
16.	Magenta
17.	DarkYellow
18.	Yellow
19.	Pink
20.	Brown

**CoPanel Object GUI Types**

<b>GUI Type</b>
IBM_3270
Workstation
HTML

***NewPanel(GUIType gui, String filename)***

Creates a new panel of the specified type. The object is created in the repository only after Commit is called. This call must be called before adding any controls to the panel.

Filename is an optional parameter used to initialize a panel from a template.

Sample script: pnl\_crt.vbs

***OpenPanel(String winName, String winLanguage, GUIType gui)***

Opens an existing panel in the repository. The default value for language is "".

Sample script: pnl\_read.vbs

***AddControl(ControlType control\_type)***

Adds a new control of the specified control\_type to the panel and returns a Control Object.

Sample script: pnl\_crt.vbs

***AddAttribute(String AttributeName)***

Adds the specified attribute to the Panel object and returns the Attribute Object. This method is called when a new attribute that is not in the current Window Model must be added to the panel.

***DeleteAttribute(String AttributeName)***

Deletes the specified attribute from the Panel object. This can be done for attributes in the Window Model and for newly added attributes.

***GetAttribute(String AttributeName)***

Returns the Attribute Object for the specified attribute.

Sample script: pnl\_read.vbs

***DeleteControl(Control ctrl)***

Deletes the control specified by the ctrl object from the panel.

***Commit(String WindowName, String Language)***

Saves the Panel object in the repository. For new panels, if a panel with the same GUI and language exists under the WindowName, an error is returned. The panel is updated if an existing panel is opened, updated and committed.

Sample script: pnl\_crt.vbs

***GetAllAttributes()***

Returns a Dictionary object containing a list of all the Attribute Objects for this panel.



### **Rollback()**

Rolls back all changes made to the panel and returns it to the NewPanel or OpenPanel state.

### **GetAllControls()**

Returns a Dictionary object with a list of all the controls in the panel.

### **SaveToFile(String PanelFileName, String HelpFileName)**

Saves the panel to a file named PanelFileName and the help to HelpFileName.

### **GetDomainValues(String Type)**

Given a Domain Type, this call returns a dictionary object containing all the values in the domain. Refer to the [Valid Domain Types and Values](#).

### **GetLanguage()**

Retrieves that language property for the current panel object.

### **SetLanguage(String Language, Boolean overwrite)**

Set the language property for the panel. If overwrite is TRUE, the panel replaces an existing panel of that language if it already exists.

### **NewLanguagePanel(String Language, String BaseLanguage)**

Create a new language panel based on an existing language panel specified by the BaseLanguage. If the BaseLanguage is a blank string, the new language will be created based on the default language.

### **Paste(VARIANT objects)**

Adds objects, supplied in panel file format, to the panel. Returns a Dictionary object with a list of the controls that were added.

### **GetDomainTypes()**

GetDomainTypes() returns a dictionary of domain type strings.

### **GetDomainValues()**

GetDomainValues() returns a dictionary of values for a given domain type.

### **TraceLevel**

Sets the trace level for the CoPanel object. The property uses the same domain values as [Trace Levels](#) in the TurboScripter Object Model. The default value is set to TRACE\_NONE. You must explicitly set the value for tracing to get trace information on the CoPanel object.

These methods are used together to provide Window Painter the valid attribute values for a given attribute. These values are contained in the Window Painter model.

See [Valid Domain Types and Values](#).

## **Trace Object - Implements Itrace**

### **OutFile**

An integer constant that specifies the name of the file to which all trace outputs should be written. If the file already exists, the existing file is deleted and a new file of the same name is opened. To save an existing trace file, get the *OutFile* property, create a *FileSystemObject* with this name and save it to another filename. The value of the *OutFile* property is initialized to the value of the key in the registry.

### **HKEY\_LOCAL\_MACHINE\SOFTWARE\BluePhoenix\AppBuilder\DWB\SCRIPTING\TURBO\_SCRIPTER\Trace\_File**

If there is an invalid value in the registry, then the *OutFile* property is assigned to *ts.out* in the current directory.

### **Level**

An integer property that specifies the severity of the messages to be written to the trace file. The following table lists the valid values:

### **Trace Levels**

---

Numeric Value	Integer Constants	Trace File Contents
0	TRACE_NONE	No trace information at all
1	TRACE_EXECUTION	Trace errors
2	TRACE_DEBUG	Trace errors and informational messages
3	TRACE_ALL	Trace detailed error and information messages Note that this value significantly impacts performance.

TRACE\_EXECUTION, TRACE\_DEBUG, and TRACE\_ALL are integer constants defined in the TSATL type library. Any other numeric value above 2 is the same as setting the value to 2.

#### **Log(String format, String param)**

This method logs messages.  
Example:

```
Trace.Log("Codegeneration %s with errors", "failed");
```

#### **SetOutFile(String newVal)**

This method opens a log file for appending.  
Example:

```
Trace.SetOutFile("output.log")
```

## Valid Domain Types and Values

Here is a list of the current domain types and values used in [GetDomainTypes\(\)](#) and [GetDomainValues\(String Type\)](#). The supported values for each type is provided in the linked subtopic.

#### **Domain Types and Values**

<a href="#">COUNTRY</a>	<a href="#">BORDER_TYPE</a>
<a href="#">COORDTYPE</a>	<a href="#">FONT</a>
<a href="#">JUSTIFICATION</a>	<a href="#">CHARTTYPE</a>
<a href="#">FORMAT</a>	<a href="#">STYLE</a>
<a href="#">COLOR</a>	<a href="#">STYLE_3270</a>
<a href="#">HSCROLL</a>	<a href="#">SELECTIONTYPE</a>
<a href="#">VSCROLL</a>	<a href="#">DRAWLINES</a>

#### **COUNTRY**

##### **Country Values**

SYSTEM	ALBANIA
ARGENTINA	AUSTRALIA
AUSTRIA	BELGIUM
BRAZIL	CANADA_ENGLISH
CANADA_FRENCH	CHINA
CZECHOSLOVAKIA	DENMARK
FINLAND	FRANCE

GERMANY	GREECE
HUNGARY	ICELAND
ITALY	JAPAN
NETHERLANDS	NEW_ZEALAND
NORWAY	POLAND
PORTUGAL	ROMANIA
SOUTH_AFRICA	SOUTH_KOREA
SPAIN	SWEDEN
SWITZERLAND	TAIWAN
THAILAND	THAILAND_BUDDHIST
TURKEY	UNITED_KINGDOM
UNITED_STATES	YUGOSLAVIA

**COORDTYPE**

**Coordtype values**

CHAR	PIXEL
------	-------

**JUSTIFICATION**

*Justification values*

LEFT	RIGHT
------	-------

**FORMAT**

*Format values*

CASEENTERED	UPPER
LOWER	FIRSTUPPER
ALLFIRSTUPPER	

**COLOR**

*Color values*

CUSTOM	DEFAULT_COLOR
BLACK	WHITE
DARKGRAY	GRAY
LIGHTGRAY	DARKBLUE
BLUE	DARKGREEN
GREEN	DARKCYAN
CYAN	DARKRED
RED	DARKMAGENTA
MAGENTA	DARKYELLOW
YELLOW	PINK

BROWN	
-------	--

## HSCROLL

### *Hscroll values*

SHOW_ALWAYS	SHOW_NEVER
SHOW_AS_NEEDED	

## VSCROLL

### *Vscroll values*

SHOW_ALWAYS	SHOW_NEVER
SHOW_AS_NEEDED	

## BORDER\_TYPE

### *Border values*

BORDER_NONE	BORDER_SIZEABLE
BORDER_DIALOG	

## FONT

### *Font values*

CUSTOM	SYSTEMFONT8
MODERN8	MODERN10
MODERN12	ROMAN8
ROMAN10	ROMAN12
ROMAN14	ROMAN18
ROMAN24	SWISS8
SWISS10	SWISS12
SWISS14	SWISS18
SWISS24	

## CHARTTYPE

### *Charttype values*

LINECHART2D	BARCHART2D
PIECHART2D	STACKEDBARCHART2D
COLUMNCHART2D	SMOOTHLINECHART2D
SCATTERCHART2D	AREACHART2D
BARLINECHART2D	HILLOWCLOSE2D
CANDLE2D	POINTANDFIG2D
LINECHART3D	BARCHART3D
PIECHART3D	STACKEDBARCHART3D

---

COLUMNCHART3D	AREACHART3D
PERBARCHART3D	BARLINECHART3D

## STYLE

### Style values

DROPDOWN	DROPDOWNLIST
SIMPLE	

## STYLE\_3270

### Style\_3270 values

DROPDOWN	DROPDOWNLIST
----------	--------------

## SELECTIONTYPE

### Selectiontype values

SINGLE	MULTIPLE
EXTENDED	

## DRAWLINES

### Drawlines values

NOLINES	VLINES
HLINES	VHLINES

## CoCtrl Object

### ***GetAllAttributes()***

Returns a Dictionary object containing a list of all the Attribute Objects for this control.

### ***AddAttribute(String AttributeName)***

Adds the specified attribute to the Panel object and returns the Attribute Object. This method is called when a new attribute that is not in the current Window Model must be added to the panel.

### ***GetAttribute(String AttributeName)***

Returns the Attribute Object for the specified attribute. Use this object only in Construction Workbench.  
Sample script: pnl\_read.vbs

### ***GetType()***

Returns the ControlType constant (defined above) that specifies this control type.  
Sample script: pnl\_read

### ***DeleteAttribute(String AttributeName)***

Deletes the specified attribute from this control. This can be done for attributes in the Window Model and for newly added attributes.

### ***GetContents()***

Returns a Dictionary Object containing the controls that are embedded in this control. Fails if no controls are present.

### ***AddControl(ControlType type)***

A control of ControlType type is added to this control.  
Sample script: pnl\_crt.vbs

#### **DeleteControl(Control ctrl)**

Deletes the Control specified by ctrl from this control.

#### **DeleteContents()**

Deletes all embedded controls in this control.

#### **ChangeParent(IcoCtrl \* old, IcoCtrl \* new)**

Changes a control objects parent.

#### **ChangeType()**

Modifies the control type.

#### **Copy()**

Creates a SAFEARRAY of BYTES that correspond to the panel file format for the control.

#### **SetDataLink( String Type, String Name )**

Creates a data link property for a control on a window. The data object is the entity specified by the Type and Name. Depending on the control type, the Type parameter should be 'VIEW' or 'FIELD', e.g. for an MCLB, it is View, while for a Column or Edit control, it is Field. The Name parameter should be the name of the data item, view or field, to be linked to the control.

Example 1: SetDataLink( "FIELD", "retrun\_code" )

This links the FIELD "return\_code" to the control. The FIELD object has to be in the scope of the Window to which the control belongs. If the data item can not be linked to the control, e.g. it is not within the window view, the method returns false.

If Name is specified as an empty string, "", the method removes any data link for the control.

Example 2: SetDataLink( "FIELD", "" ) - This empties a data link.

This method can link to a data item within a sub-view of the window view. However, if there are multiple instances of that data item within the window view, this method uses the first instance found. A subsequent fix is planned to allow a specific instance of the data item to be linked to the control.

### **Attribute Object**

#### **set\_Value(VARIANT val)**

Sets the Value of this attribute to that specified by val. All variables in VBScript are of type VARIANT, which is a generic type for all other supported types. This method would be successful as long as the value is of the right type, for example, String, Integer etc.

Sample script: pnl\_crt.vbs

#### **get\_Value()**

Returns the value of the attribute. If the attribute is a compound attribute, like Font or Color, then this value would contain an object like Font or Color.

Sample script: pnl\_crt.vbs

#### **get\_Name()**

Returns the name of this attribute.

Sample script: pnl\_read.vbs

#### **DeleteAttribute()**

Removes this attribute from the parent control.

#### **Font Object and Color Object**

These are compound attributes and can be treated like a sub-control, since they contain the same methods that characterize a control object. They have attributes that can be accessed through GetAllAttributes or GetAttribute(String AttributeName).

## **Starting TurboCycler**

TurboCycler is an installation component of the AppBuilder program. You can access TurboCycler from the following application development tools:

- Database Diagrammer
- Entity Relationship Diagrammer
- Matrix Builder
- Process Dependency Diagrammer
- State Transition Diagrammer
- Window Flow Diagrammer

To open the TurboCycler from the Construction Workbench, select the object in the Hierarchy window to which you want to apply the TurboCycler script and select *Tools > TurboCycler*.

The following topics are discussed in this section:

- [Selecting Generation Templates](#)
- [Setting Up Generation Templates](#)
- [Generating Your Application](#)

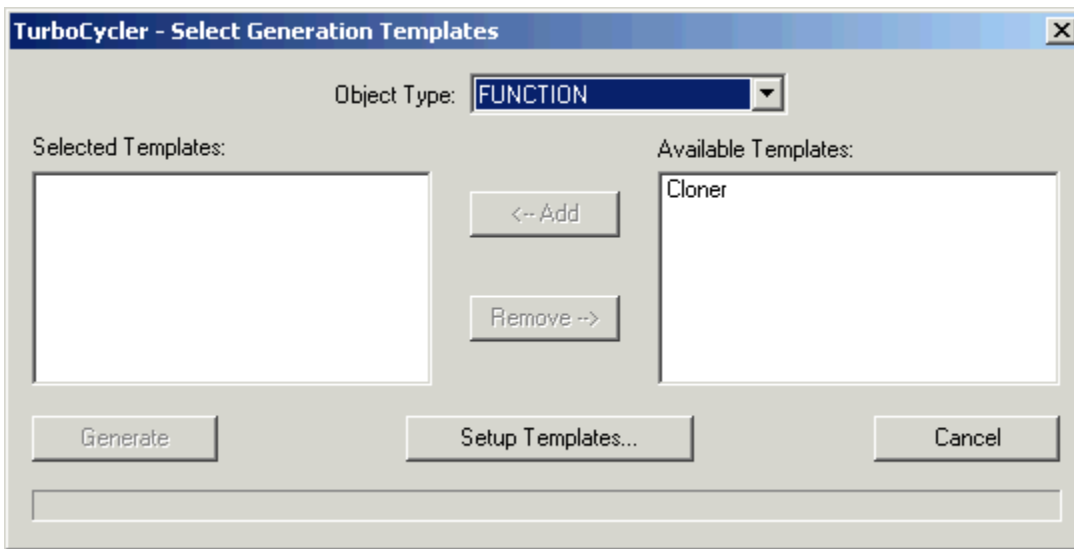
## Selecting Generation Templates

To select a TurboCycler generation template, complete the following steps:

1. From Windows® Start menu, select **All Programs > AppBuilder > Construction Workbench**.
2. Open a project in the Hierarchy and select an object that you want to use the TurboCycler on. Select *Tools > TurboCycler*.

The TurboCycler - Select Generation Templates window opens.

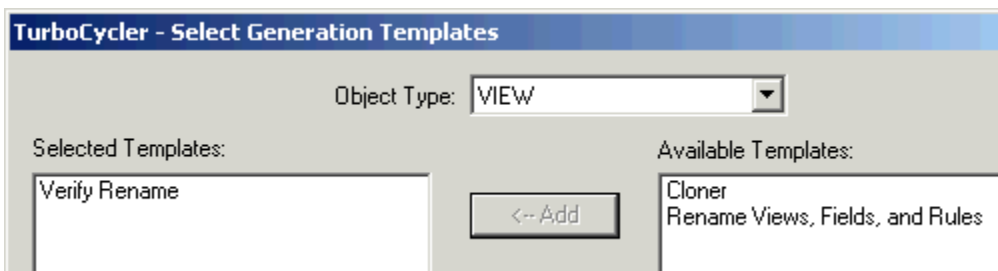
### TurboCycler-Select Generation Templates window



The Selected Templates and Available Templates list boxes display templates that you can use with the selected object type (in [TurboCycler-Select Generation Templates window](#), the object type is Function).

3. Select the templates that you want to use with the *Add* and *Remove* buttons.

### Selecting a Template in TurboCycler



If you add additional templates, alter the default templates, or create new templates (using the TurboCycler Developer's Kit), the objects you

create are available through the Query option.

4. Select *Generate* , *Setup Templates*, or *Cancel* . *Generate*.

5. *Generate* starts the generation process. *Setup Templates* displays a window in which you can select the modules to generate (see [Setting Up Generation Templates](#)). *Cancel* returns you to the previous window.

## Setting Up Generation Templates

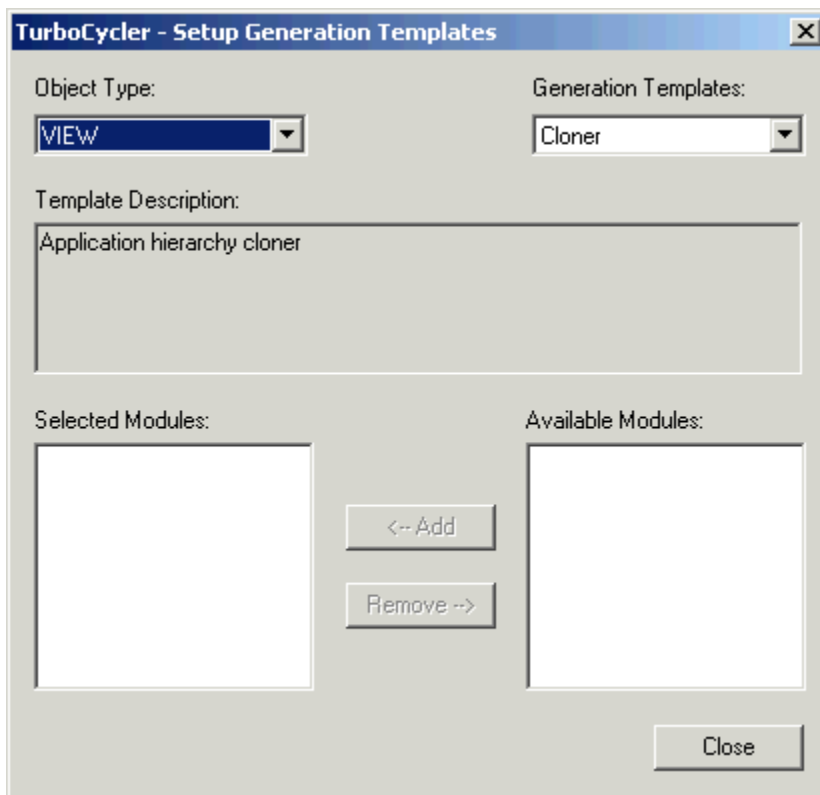
To set up your generation template, use the following procedure.

### Procedure - Setting up Templates

1. Select **Setup Templates** .

The TurboCycler Setup Generation Templates dialog appears. Use this dialog to specify precisely what output you want TurboCycler to generate (for example, whether the application should perform an SQL update or an SQL insert, or both) The output choices are called modules.

#### Generation Template Setup Window

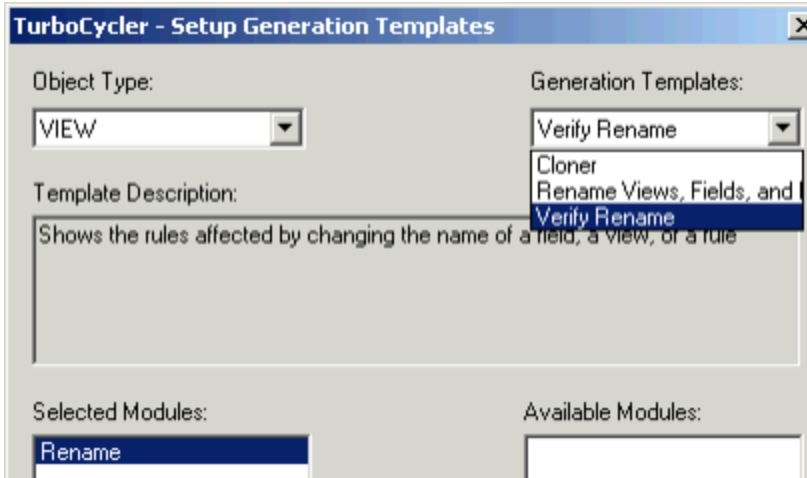


The Object Type list box displays the repository object type, and the Generation Templates list box lists the available templates for that object type ([Generation Template Selection](#)). The text in the Template Description box describes the selected template. The Selected Modules and Available Modules fields identify the available modules for this object type/template combination.

2. Select the modules to generate with the *Add* and *Remove* buttons.

#### Generation Template Selection





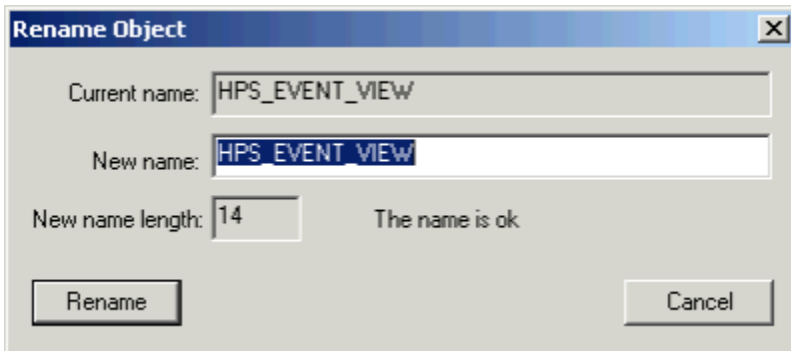
## Generating Your Application

When you have selected the appropriate modules, you can generate the application. To generate an applications, complete the following steps:

1. Select the **Generate** button in the Select Generation Templates window to start the TurboCycler process.

The Rename Object window opens.

### Rename Object Window



During the generation process, a status window notifies you of the progress of the generation.

2. Click *Cancel* to stop TurboCycler between modules.

The program returns to the Construction Workbench TurboCycler window ([TurboCycler-Select Generation Templates window](#)). You can review the results of the TurboCycler process after completion.



The TurboCycler process generates objects using names defined by the templates for the objects you select. If you run TurboCycler more than once, previously generated objects with modifications are replaced by newly generated objects. Because of this, you must carefully examine the objects created by a generation. To validate the generation results and make sure that no desired user modifications have been lost, examine the hierarchy, rule, and window objects. If the results of a generation process are not satisfactory, you can roll back the session changes from the Construction Workbench. Click *File > Rollback* from the Construction Workbench.

## Using Default Templates

This topic includes:

- [Understanding Default Templates](#)
- [Developing the Application](#)
- Information about using default templates:
  - [CRUD Rules Template](#)
  - [GUI Rules/Windows Template](#)
  - [Utilities Template](#)
  - [Rename Views, Fields and Rules Template](#)
  - [Rename Verify Template](#)
  - [Hierarchy Cloner Template](#)

## Understanding Default Templates

TurboCycler software includes a set of six default templates. You can generate a fully functional reference table application using all of the six templates. The default templates specify part of the TurboCycler output and make modifications easier. The default templates are designed for single-byte characters. For templates using the 'String Manipulation function' in a DBCS environment, you must modify the default templates. See [Editing and Compiling a Template](#) for more information.

- [CRUD Rules Template](#): includes SQL statements (Create, Read, Update, and Delete database access operations)
- [GUI Rules/Windows Template](#): supplies graphical interface event-handling procedures
- [Utilities Template](#): contains message-handling procedures used by generated objects
- [Rename Views, Fields and Rules Template](#): modifies the source of any rules that refer to a renamed Field or View object
- [Rename Verify Template](#): shows the rules in the repository that must be modified if the name of the selected object is changed
- [Hierarchy Cloner Template](#): creates a new Hierarchy of repository objects based on an existing Hierarchy

The TurboCycler process uses the default templates to create output objects with fully coded rules including calls to system components as needed, painted windows with predefined styles, and view and field physical data structures. These objects are automatically related in a logical hierarchy in the repository.



Two or more users cannot run TurboCycler templates that create and update the same repository objects. This is true for objects that are sent through TurboCycler as well as other objects that the template might access to update or create. If two or more users attempt to run TurboCycler templates that create and update the same repository objects, the first user locks the object, and the second user gets an error message "Object exists in the repository in an Uncommitted State."

[Using Default Templates](#) describes how to use the default templates. It details the output objects that the templates generate and defines the specific Rules, Windows, and Hierarchies generated by each default template.

The source code for each of these templates is included in the TurboCycler Developer's Kit. Two source code samples are included in [Template Samples](#).

Using the default templates, you can generate a functional application, including complete function hierarchies, detail and list windows, database access rules, and standard utilities. You can write generation templates to automate an unlimited number of processes in the development lifecycle. You can also write templates to customize processes like forward engineering, transformation, reverse engineering, and any other process that generates objects in the repository from other repository objects.

## Developing the Application

Before starting TurboCycler, perform the following Construction Workbench modeling steps:

1. Create an entity relationship diagram (ERD).

This establishes the ERD as a logical model that consists of entities, relationships, properties, identifiers, and data types. See [Defining an Entity Relationship Diagram](#) for more information.

2. Forward engineer the ERD into a database diagram (DBD).

Forward engineering creates a relational model containing tables, keys, and columns. See [Forward Engineering an Entity Relationship Diagram](#) for more information.

3. Transform the DBD into a Hierarchy Diagram with file structures. See [Transforming a Database Diagram \(DBD\)](#) for more information.

For detailed information about creating an ERD, forward engineering an ERD, and transforming a DBD, refer to the *Developing Applications Guide* and the *Development Tools Reference Guide*

These steps populate the repository with the information that TurboCycler needs to generate a reference-table application. Without TurboCycler, developing applications using these objects requires you to write rules code yourself to reflect the model.

### Encountering Errors

If TurboCycler encounters a repository error or other errors that are specific to an object, a message is displayed.

In the error message window, select Yes for TurboCycler to continue processing other objects while omitting all objects that encountered the error.

### Session Reports

The TurboCycler Report option displays a summary of the last TurboCycler session. This file is overwritten for each TurboCycler session. If you want to save the session information, then the file *TC.OUT* needs to be saved manually. The location for this file can be specified on the TurboCycler set of options of the Workbench Options dialog ( *Tools > Workbench Options* , then click *Script > TurboCycler* ).

## CRUD Rules Template

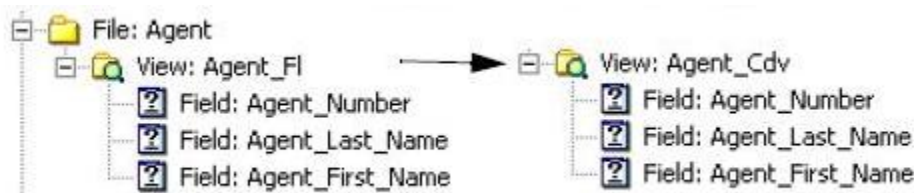
You can generate Create, Read, Update, Delete rules (CRUD) for any existing file hierarchy in the repository. Because of the traceability information stored in the repository during forward engineering and transformation, four object types in the repository (Files, Tables, Entities, and many-to-many relationships) can generate CRUD rules. If you want to generate CRUD rules from non-file objects, you must perform the appropriate forward engineering and transformation steps before the generation process, or an error message is displayed. The CRUD Rules template contains six modules:

- [Clone Data View Module](#)
- [SQL Delete Rule Module](#)
- [SQL Fetch Rule Module](#)
- [SQL Insert Rule Module](#)
- [SQL Select Rule Module](#)
- [SQL Update Rule Module](#)

### Clone Data View Module

Clone Data View creates a view in the repository. The view name is the name of the file (for long file names, truncated to a maximum of 19 single-byte characters or eight DBCS characters), plus the suffix *\_CDV* . This view contains, as its children, all of the fields included in the file data view. Assorted CRUD rules and some GUI rules and windows use the cloned data view to avoid problems of hierarchy ambiguity. The following figure shows a generated clone data view and the corresponding file data view.

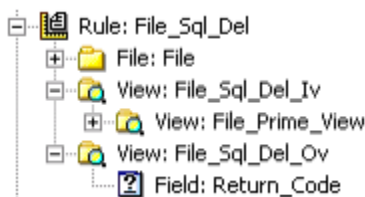
#### Generated Clone Data View with its Corresponding Data View



### SQL Delete Rule Module

SQL Delete Rule creates a Rule with appropriate Input View, Output View, and Rules Code that handles record deletion in the table specified by the file implementation name. It determines which file to delete the record from by the primary key view mapped to its input view. An SQL code is returned from the delete operation. The following figure shows an example of a delete rule hierarchy.

#### Delete Rule Hierarchy



### SQL Fetch Rule Module

SQL Fetch Rule creates a rule with appropriate input view, output view, and rules code. This rule fetches records from the table specified by the file implementation name. The fetch rule is compatible with smooth scrolling techniques through the use of the *NUMBER\_OF\_RECORDS* field in the input view. For each fetch, fifty occurrences of records are fetched according to the parameters mapped to the primary key view, cloned data view, and *FETCH\_TYPE* field in the rule input view.

You can specify the *FETCH\_TYPE* as blank, which defines a fetch for all records whose key is greater than or equal to the key mapped to the input view, or as "D", which defines a fetch using a LIKE clause against all fields in the key. The fetch rule returns and maps the SQL code to the field *RETURN\_CODE* if an SQL error occurs. The number of records fetched is returned to the *NUMBER\_OF\_RECORDS* field and all fetched records are returned to the occurring cloned data view in the output view.

### SQL Insert Rule Module

SQL Insert Rule creates a rule with appropriate input view, output view, and rules code. This rule inserts a record in the table specified by the file

implementation name. It inserts the record mapped to the cloned data view of its input view and returns the SQL code from the insert operation.

### **SQL Select Rule Module**

SQL Select Rule creates a rule with appropriate input view, output view, and rules code. This rule selects a record by primary key from the table specified by the file implementation name. It uses the primary key mapped to its input view to select a single record in the database table and returns the contents of the record in the cloned data view of the output view. It returns the SQL code from the select operation.

### **SQL Update Rule Module**

SQL Update Rule creates a rule with appropriate input view, output view, and rules code. This rule handles updates to a database record selected by the primary key in the table specified by the file implementation name. It sets all fields of the record of the table to the values mapped to the cloned data view of the rule input view. It returns the SQL code from the update operation.

## **GUI Rules/Windows Template**

You can generate GUI rules and windows for any forward engineered and transformed entity hierarchy in the repository. The GUI Rules/Windows template extracts information already defined in the logical, relational, and structural models of the repository. This template uses traceability information defined during forward engineering and transformation to generate GUI rules and windows from entities, many-to-many relations, tables, and files in the repository. If any object of these types does not have the required traceability information, you cannot use this template to generate GUI rules and windows for that object, and an error message is displayed.

The GUI Rules/Windows Template contains the following modules:

- [Detail Display Module](#)
- [Display Driver Module](#)
- [Browse Display Module](#)
- [List Display Module](#)
- [Query Display Module](#)
- [Logical Driver Module](#)

### **Detail Display Module**

Detail Display creates both a rule and a window for each entity that you specify, with appropriate hierarchies that have references to sets and components. This rule and window display data details in the application interface. They display information and allow standard record maintenance (create, read, update and delete). You can edit all fields in the data view of the detail display window.

The hierarchy generated for a kernel entity is different from the one generated for a non-kernel entity. The detail display for a kernel entity is a choice in the AppBuilder Application Execution menu for accessing the generated application; therefore, a function and process are generated for every kernel entity. Because the detail display rule for a kernel entity is the root rule for the generated process, it contains no input or output view. Use the Open choice in the File menu to use a kernel entity detail display to select an instance of the kernel entity.

Because the details of a non-kernel entity are meaningful only in the context of another entity, you can navigate to a non-kernel detail display only from another entity; it does not serve as an entry point into the application. For this reason, the detail display rule for a non-kernel entity contains an input and an output view. Also, you cannot open an instance directly from a non-kernel detail display, as it is necessary to clarify its context through navigation from another object.

A generated detail display window always contains File, Edit, and Options menus. Use the File menu to create a new object, save an object, delete an existing object, or exit the application. The Edit menu provides undo and redo facilities. Use the Options menu to eliminate the status bar that appears at the top of the detail display.

View or Query menu choices are available when the table implementing the entity contains referred-by keys (for the View menu) or foreign keys (for the Query menu). You can use View and Query menus to navigate from object to object (entity to entity) in the application. The status line under the menu bar displays information about current actions.

### **Display Driver Module**

Display Driver creates a rule with appropriate hierarchies that have references to sets and components. This rule initializes window parameters and enables or disables objects on the detail window by interacting with the Detail Display rule. The display driver rule does not have an output view, so no return code is passed back to the calling rule.

### **Browse Display Module**

Browse Display creates both a rule and a window with appropriate hierarchies that have references to sets and components for any kernel entity. This rule and window provide a way of looking at information for a kernel entity without updating the information. You can use this browsing technique for object-to-object navigation only for kernel entities. A detail display rule for a non-kernel entity handles object-to-object navigation. Use the browse display also when navigating to a kernel entity to avoid recursion in the repository.

The browse display window is very similar to a detail display window. However, the browse display offers only the Exit choice in the File menu. You can perform no other actions including object-to-object navigation. For that reason, there is no status line.

### **List Display Module**

List Display creates both a rule and a window with appropriate hierarchies that have references to sets and components for the entity. Use this window to select an instance of this entity from the Query menu. The list display rule shows a list of entity instances that meet the criteria that the primary key specifies. It supports smooth scrolling, so you query the table and select a single instance in the table implementing this entity. The list display rule takes as input the primary view of the entity that it uses as search criteria. The output of the list display rule is a single instance of the cloned data view.

The list display window contains a multicolumn list box with one column for every field in the data view of the file corresponding to this entity. The top of the window has an edit field for every field in the primary view that you can use to specify selection criteria for a subset of entity. The menu contains a single choice for toggling the status line on and off. Use the push buttons at the bottom of the window to:

- Select a set of rows in the list box according to the criteria specified at the top of the window
- Select a single row from the list box
- Cancel from this window

The status line under the menu bar displays information about current actions.

### Query Display Module

Query Display creates both a rule and a window with appropriate hierarchies that have references to sets and components for the entity. Use this rule and window to navigate to this entity from another entity, usually through a View menu. The query display rule shows all entity instances that meet the criteria specified in the primary key. It supports smooth scrolling, so you can specify information to query the table and select a single instance in the table implementing this entity.

If you select an instance of this entity, either the detail display for this entity for a non-kernel entity or browse display for a kernel entity is conversed. The query display rule takes as input the cloned data view of the entity, which it uses as search criteria if you specify any. The query display rule is for navigation and viewing only.

The query display window has a multicolumn list box with one column for every field in the data view of the file corresponding to this entity. Above the multicolumn list box is an edit field for every field in the primary view. Use it to specify search criteria for a subset of entity instances. The menu contains a single choice for toggling the status line at the top of the window on and off. Use the push buttons at the bottom of the window to:

- Select a set of rows in the list box according to criteria at the top of the window
- Select a single row from the list box
- Cancel from this window

The status line under the menu bar displays information about current actions.

### Logical Driver Module

Logical Driver creates a rule with appropriate hierarchies that have references to sets and components that acts as a layer between the detail display and browse display presentation rules and the SQL rules. Logical Driver performs all complex field-to-field mappings when making calls to SQL rules. It takes the action that is to be performed on the instance along with the entity instance (the cloned data view) in the input view. If errors occur, it automatically calls the messaging routine defined by the Message Display template.

### Utilities Template

The Utilities template generates a message-handling rule and window for almost any modal message that a rule might need, such as an SQL failure. The only module included in this template is the Message Display Module. Message Display creates the HPS\_MODAL\_MESSAGE rule and window with appropriate hierarchies that have references to sets and components to display any four-line message. You can specify the push buttons to display through the HPS\_MESSAGE\_TYPERES. The output view returns your response to the message.

### Rename Views, Fields and Rules Template

Whenever you must change the name of an existing FIELD or VIEW object, use the Field and View Rename template to automatically modify the source of any rules that refer to the renamed object.

To change the source of rules that refer to a renamed object, complete the following steps:

1. Go to Construction Workbench.
2. Select a single FIELD or VIEW object to rename.
3. Select *Tools > TurboCycler* and the *Rename Views, Fields and Rules* template.
4. When the template displays the Rename window, type the new name for the object.

It automatically modifies the source of any rules that refer to the object to use the new name. Before finishing, the template shows a list of the rules it modified.

The Status pane displays the generation process.

5. Select *Tools > Stop TurboCycler* to stop TurboCycler generation.

### Rename Verify Template

The Rename Verify template does not modify any rules or change the name of the selected object. It only shows which rules in the repository need to be modified if the name of the selected object is changed.

When you rename views that have a window parent, the window panel is not modified at all. The rule source change is a blind search and replace.

### Hierarchy Cloner Template

The Hierarchy Cloner template creates a new hierarchy of repository objects based on an existing hierarchy. The new hierarchy consists of a

mixture of both newly created objects and existing objects from the source hierarchy. Using this template involves the following:

- [Understanding the Cloning Process](#)
- [Using the Hierarchy Cloner](#)
- [Setting Initialization Parameters](#)

For each object in the source hierarchy, cloner relies on pairs of search and replace strings to determine whether it should reuse the existing object or create a new object based on the existing object.

The Hierarchy Cloner never modifies or has any effect on existing objects in the repository with one exception: while creating the new hierarchy, cloner can establish a parent/child repository relation between a newly created object and an existing object.



When cloning hierarchies containing error, lookup, or define sets, the encoding and display attribute values are replaced blindly without error checking for value overflow and such. Make sure that these values do not overflow and are valid.

### Understanding the Cloning Process

This section discusses how the cloning process works. Each object that the cloner examines must meet the following conditions to be cloned:

- The name of the object must be changed after cloner does string replacement using the strings you supply.
- The type of the object must be in the user-specified list of types to clone.

If either of these conditions is not met, the object is reused, and its child objects are not examined.

If the object is to be cloned, the cloner creates a new object of the same type with the cloned name and copies certain properties from the source object to the cloned object. The new object is attached to the new object hierarchy, and cloning continues by examining children of the source object. If the cloned object already exists in the repository, it is reused and never modified.

The properties copied are as follows:

- Those that are available in TurboCycler
- Writable
- Not required to be unique by the repository

Thus, the template supplied is general purpose in nature and is not geared specifically to any one type of application hierarchy. If you have special requirements for cloning, customize the supplied template using the TurboCycler Developer's Kit.



Rollback does not remove all unwanted objects after cloning by TurboCycler.

### Using the Hierarchy Cloner

Use the Hierarchy Cloner template from the Hierarchy Diagrammer to clone application hierarchies and to make copies of an existing hierarchy. To do this, you must identify one or more substrings contained in the names of the existing objects that can be used to guide the cloning process.



If you clone a window with cloning fields, you will lose the link properties of all the objects on the window panel.

The following objects can be cloned:

- Function
- Process
- Rule
- File
- View
- Field
- Set
- Value
- Component
- Window
- Physical event
- Report
- Section
- Component folder
- Bitmap

In addition to these, the cloner reuses BITMAP images, but it does not clone them. COMPONENT FOLDER does not copy or clone external tab information, and REPORT does not copy or clone the content of the Report. The cloner copies a component source from the original; it does not clone a component source to a new component.



The information in the Description field, except for COMPONENT and COMPONENT FOLDER, is saved to the repository. Description field is copied, not cloned.

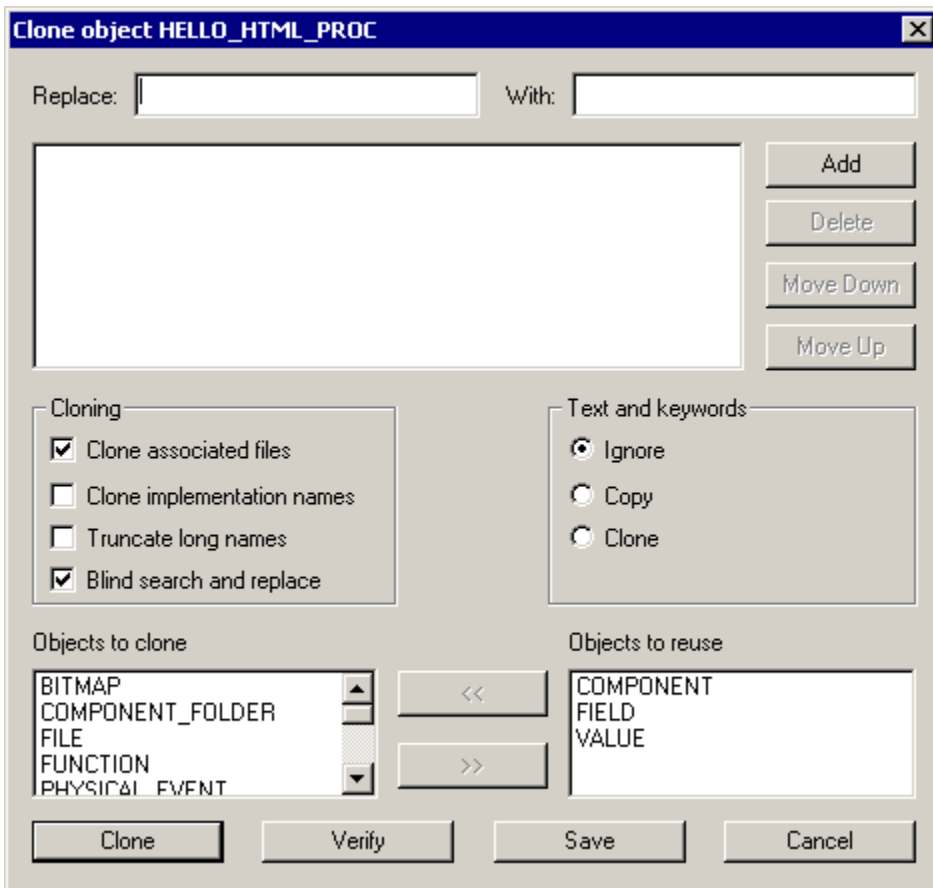
Because the cloner never modifies existing objects in the repository, subsequent regeneration of this template has no effect.



Always select a single object and choose *TurboCycler Selected* from the Analysis menu. Cloner automatically examines all objects under the selected object. Do not pass multiple objects for cloning.

When cloner executes, a dialog is displayed. Use this dialog to specify settings for the cloned object.

#### Clone dialog example



The controls and their meanings are listed in the following table:

#### Cloner Controls and Descriptions

Control	Description
Replace (entry field)	The substring to search for
With (entry field)	The replacement string to use
Add (push-button)	Add the new pair of strings to the list
Delete (push-button)	Delete the selected pair of strings from the list
Move Up (push-button)	Move the selected pair of strings up in the list
Move Down (push-button)	Move the selected pair of strings down in the list

Objects to clone (listbox)	Objects listed here are candidates for cloning
Objects to reuse (listbox)	Objects listed here will never be cloned
Clone (push-button)	Begins the cloning process
Verify	Once settings have been specified above, (Replace XXX With YYY>Add), the Cloner verifies what will be modified When you do the cloning.
Save	Settings are saved to Turbo.ini
Cancel	Cancels the process and closes the dialog.

When cloner is finished, the new hierarchy is displayed. In this display, you can browse the new hierarchy and see the objects that were created.

### Setting Initialization Parameters

The settings entered in the cloner dialog are saved to the CLONER section in the TURBO.INI. This file contains the default settings for the Cloner dialog ([Clone dialog example](#)):

```
[ CLONER ]
REPLACE=EMPLOYEE COMPANY FIRM
WITH=MANAGER ORGANIZATION COMPANY
TEXT_AND_KEYWORDS=IGNORE
SOURCE_FILES=YES
REUSE=FIELD VALUE COMPONENT
IMPLEMENTATION_NAMES=NO
TRUNCATE_LONG_NAMES=NO
```

The REPLACE entry contains a series of substrings to search for, separated by spaces. The WITH entry contains one replacement string for each substring in the REPLACE entry. TEXT\_AND\_KEYWORDS may be set to:

- IGNORE: created objects have no text or keywords
- COPY: created objects have the same text and keywords as the source object
- CLONE: created objects have the same text and keywords as the source object except modified by the name cloning procedure

If SOURCE\_FILES is NO, then created objects have identical source files as the source object; if it is YES then source files are cloned. The REUSE entry contains a list of object types separated by a single space that are never to be cloned, regardless of their names. The IMPLEMENTATION\_NAMES entry can be either YES or NO depending on whether implementation names and screen literals should be cloned. When cloner forms a name that is too long, it either automatically truncates the name or displays a dialog asking you to provide a shorter name. This choice is controlled by the TRUNCATE\_LONG\_NAMES entry, which can be either YES or NO.

### Clone Implementation Names

When you clone a Field, if the *Clone Implementation Names* option is not checked, the implementation name is cloned anyway, and other properties such as screen literals are left unchanged. However, if you check *Clone Implementation Names*, the implementation name and the screen literals are cloned. The reference table name is never cloned.

## TurboCycler Tutorial

TurboCycler software complements and enhances the AppBuilder environment. Because of their compatibilities, the sample TurboCycler tutorial is an extension of the AppBuilder environment walkthroughs in the workstation Workbench reference.

The tutorial summarizes the automobile rental agency sample from AppBuilder training and shows the results when you use the TurboCycler product with the same repository objects. The tutorial that follows assumes that you perform all of the steps for each tool.

If you imported the IVP project into the repository, the ERD and the entities mentioned in this tutorial are all available, so you don't have to create them from scratch.

The steps are:

- [Reviewing Forward Engineering and Transforming](#)
- [Generating with TurboCycler](#)

### Reviewing Forward Engineering and Transforming

This section describes the TurboCycler building process. Refer to the AppBuilder *Developing Applications Guide*, *Development Tools Reference Guide*, and online help for greater detail about building the models the TurboCycler walkthrough uses.

- [Defining an Entity Relationship Diagram](#)
- [Forward Engineering an Entity Relationship Diagram](#)



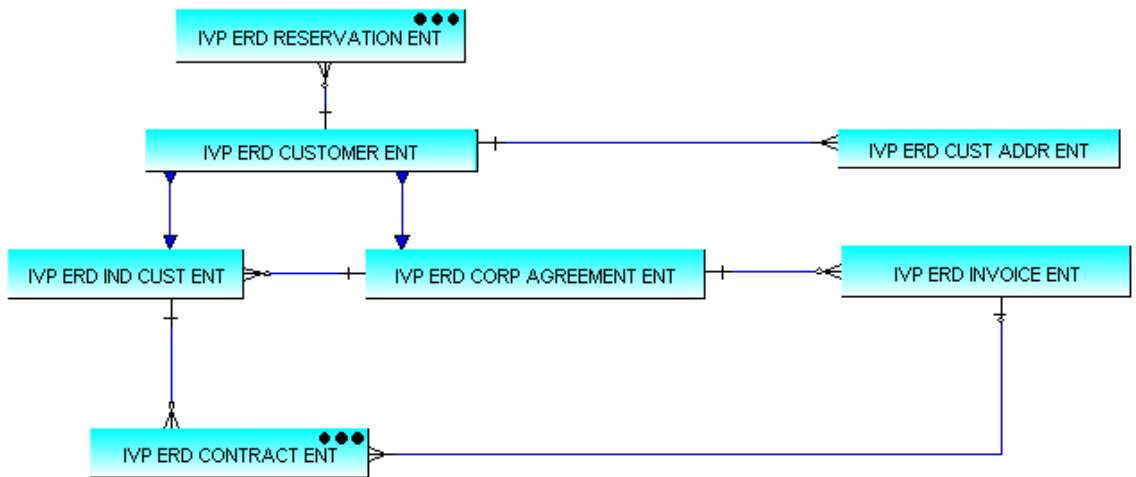
- [Transforming a Database Diagram \(DBD\)](#)

### Defining an Entity Relationship Diagram

The first step in designing your application is to create an Entity Relationship Diagram (ERD). To create an ERD, complete the following steps:

1. In the Construction Workbench, create a new Entity Relationship Diagram ( **File > New > Entity Relationship Diagram** ).
2. Proceed to create the drawing shown in the following figure:

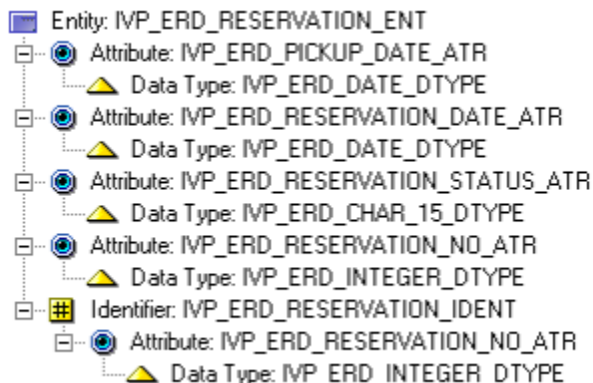
#### Entity relationship diagram



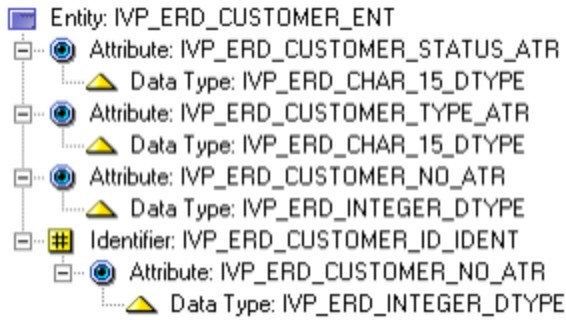
3. Select *Select All* in the Edit menu; then, select *Open as Hierarchy* in the Edit menu to place the objects in the Repository tab in the Workbench.

4. Create the attributes, identifiers, and data types for the IVP\_ERD\_RESERVATION\_ENT, IVP\_ERD\_CUSTOMER\_ENT, IVP\_ERD\_IND\_CUST\_ENT, IVP\_ERD\_CORP\_AGREEMENT\_ENT, IVP\_ERD\_CUST\_ADDR\_ENT, IVP\_ERD\_INVOICE\_ENT, IVP\_ERD\_CONTRACT\_ENT entities, as shown in the following figures.

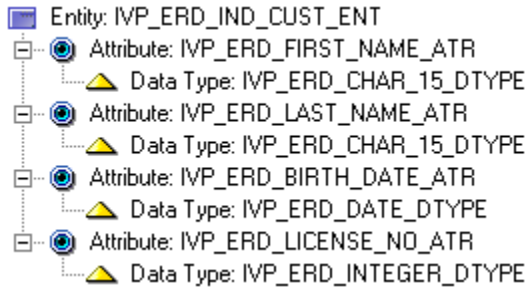
#### The IVP\_ERD\_RESERVATION\_ENT entity



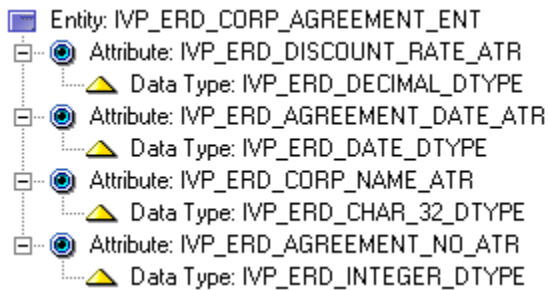
#### The IVP\_ERD\_CUSTOMER\_ENT entity



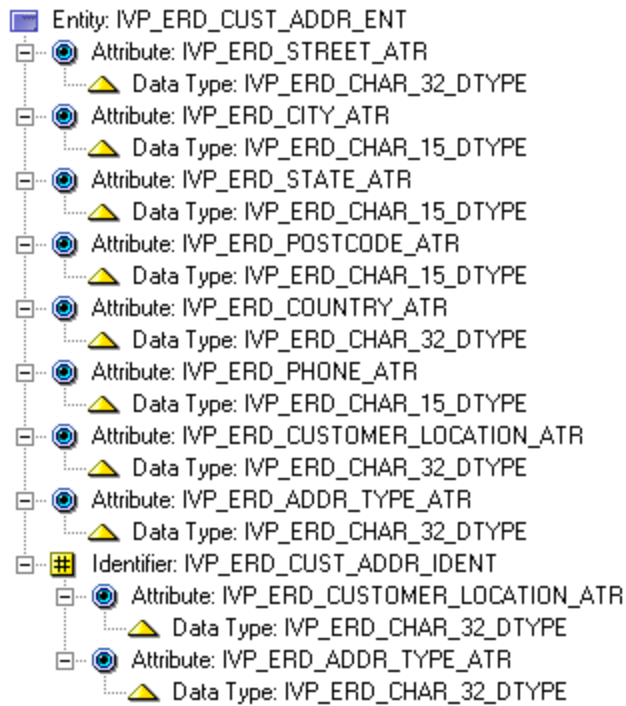
**The IVP\_ERD\_IND\_CUST\_ENT entity**



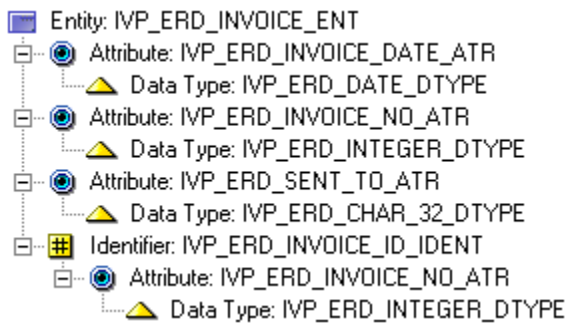
**The IVP\_ERD\_CORP\_AGREEMENT\_ENT entity**



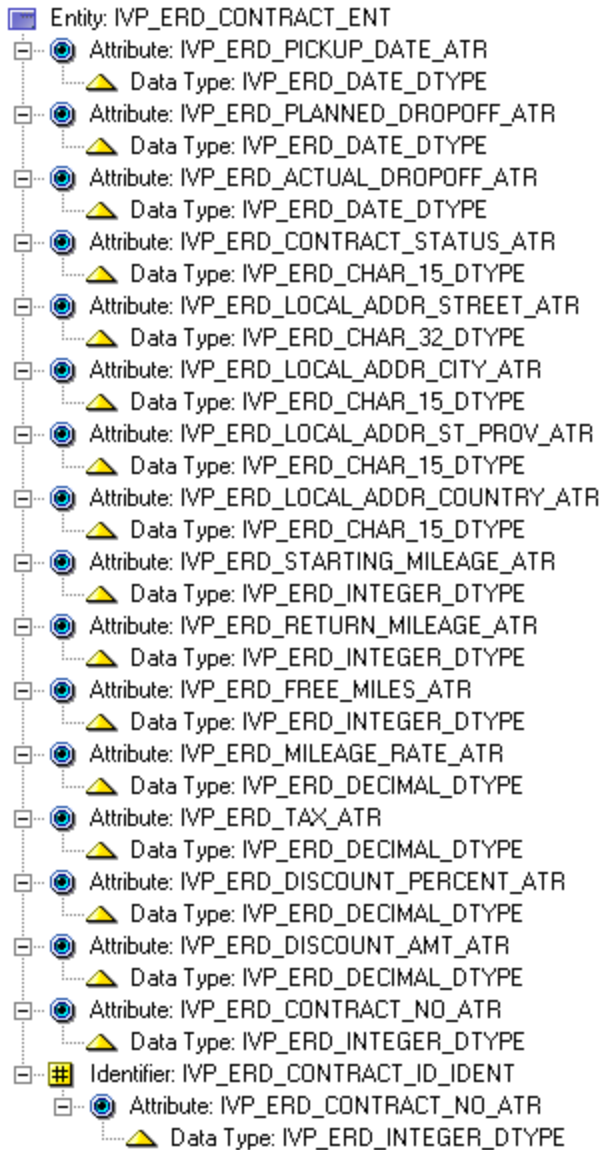
**The IVP\_ERD\_CUST\_ADDR\_ENT entity**



#### The IVP\_ERD\_INVOICE\_ENT entity



#### The IVP\_ERD\_CONTRACT\_ENT entity



5. In the Construction Workbench, select *File > Commit*, press **Ctrl+M**, or click the *Commit* toolbar button. Name the entity relationship diagram IVP\_ERD\_EX\_RESERVATION.

### Forward Engineering an Entity Relationship Diagram

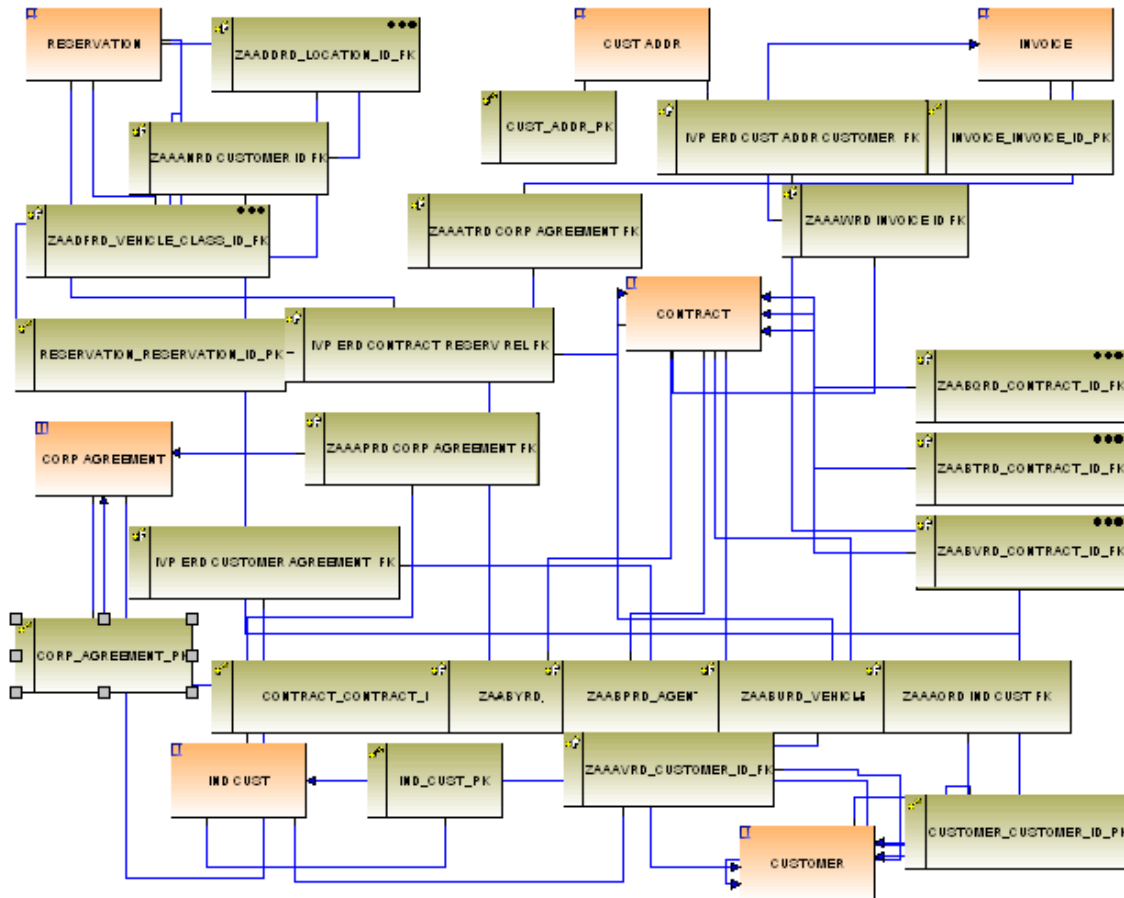
This section assumes you have created the Entity Relationship Diagram (ERD) and attributes described in [Defining an Entity Relationship Diagram](#).

1. Select the ERD that you just created.
2. Verify that each relationship has a cardinality symbol at each end.
3. Choose *Forward engineer* in the *Analysis* menu. Repeat the previous steps if you have errors.
4. Create a new Database Diagram (DBD) to view the tables created and their relationships:
  - Select **File > New and select Database Diagram**. A blank DBD opens.
  - In the new DBD, add a Table object and then double-click the table object. This opens the Insert Table dialog.
  - Use the Query button to see the tables that have been created and add each of the tables to the DBD.
  - To show all objects and their relationships, select each of the tables in turn and select *Edit > Explode* (or F8). This will display the tables, the primary keys, the foreign keys, and the relationships between the entities.



You might have to rearrange the exploded objects so that they are displayed in an orderly and convenient manner.

### Tables and keys that forward engineering created



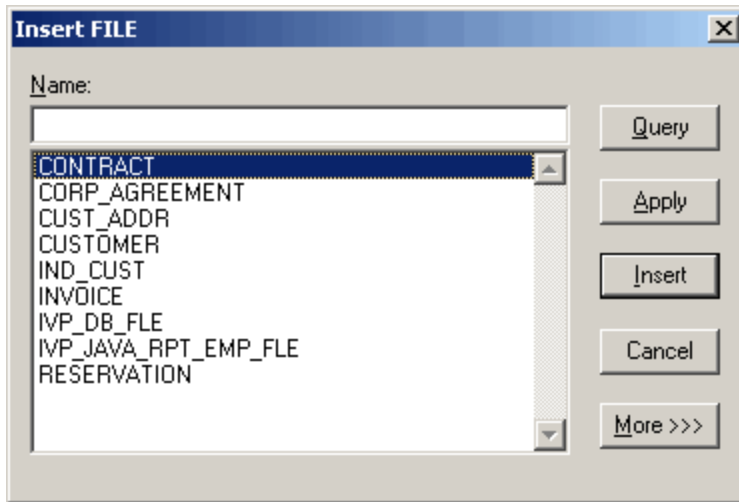
5. Select *File > Commit* , press Ctrl+M, or click the *Commit* icon. If you have displayed the columns and keys in the DBD, name and save the drawing.

### Transforming a Database Diagram (DBD)

This section assumes that you have created the Entity Relationship Diagram (ERD) and attributes in the ERD as described in [Defining an Entity Relationship Diagram](#) and that you forward engineered as described in [Forward Engineering an Entity Relationship Diagram](#). The following steps populate the repository with information that TurboCycler needs to generate a reference-table application with Views, Rules, and Windows needed for any application program. To transform a DBD, complete the following steps:

1. Select **File > Open > Database Diagram** to open the DBD that you have created.
2. Be sure that all tables display in the Database Diagram.
3. Select **Transform** in the *Analysis* menu.  
The results of the transformation are displayed in the Analysis tab of the Output window.
4. Right-click *Repository* and select **Insert > File** .
5. Query for the file entities created.

### Insert File window



6. Insert the files into the Hierarchy. You can select several files at a time, then you can click *Insert*. If you insert one file at a time, click *Apply* between each selection and click *Insert* when you want to close the Insert File window.

## Generating with TurboCycler

The following walkthrough shows how to complete the TurboCycler process using default templates. The steps are:

- [Editing Screen Literals](#)
- [Selecting TurboCycler Windows](#)
- [Preparing the Application](#)
- [Procedure - Testing the Application](#)

### Editing Screen Literals

Transformation generates default screen literals that might require editing. Names are system generated or concatenated from labels that you entered earlier. The windows generated by TurboCycler benefit from literals edited for brevity, conciseness, or appearance. Rename screen literals before starting TurboCycler generation.



If you do not edit literals once now, you will need to edit at least four windows later.

To change the screen literal, complete the following steps:

1. Review each field and display the Object Property window for each field. If it is not already displayed, right-click the field and choose Properties.  
The Object Property window displays as follows.

### Properties Field window to alter screen literals

Name	Value
[-] General (FIELD)	
Name	CUSTOMER_NO
Transformation Status	N/A
Preparation Status	N/A
System ID	ZAAVLE
Field Picture-Storage	
Field Picture-Display	
Screen Literal-Long	CUSTOMER_NO
Field Format	Small Integer or Integer
Field Length	00015
Field Fraction	00
Range-Minimum Value	
Range-Maximum Value	
Reference Table Name	
Screen Literal-Short	CUSTOMER_NO
Implementation Name	CUSTOMER_NO
[+] Audit	
[+] Remote Audit	
[-] Relationship [VIEW_INCLUDES]	
Relationship type	VIEW_INCLUDES
Parent name	CUSTOMER_CUSTOMER_ID_PK
Child name	CUSTOMER_NO
Separator ID	0
Sequence number	10
Occurs times	
Null indicator	Not Null
Null Default Value	
[+] Relationship Audit	
[+] Relationship Remote Audit	
<b>Properties</b>   Text   Keywords   RelText   RelKeywords	

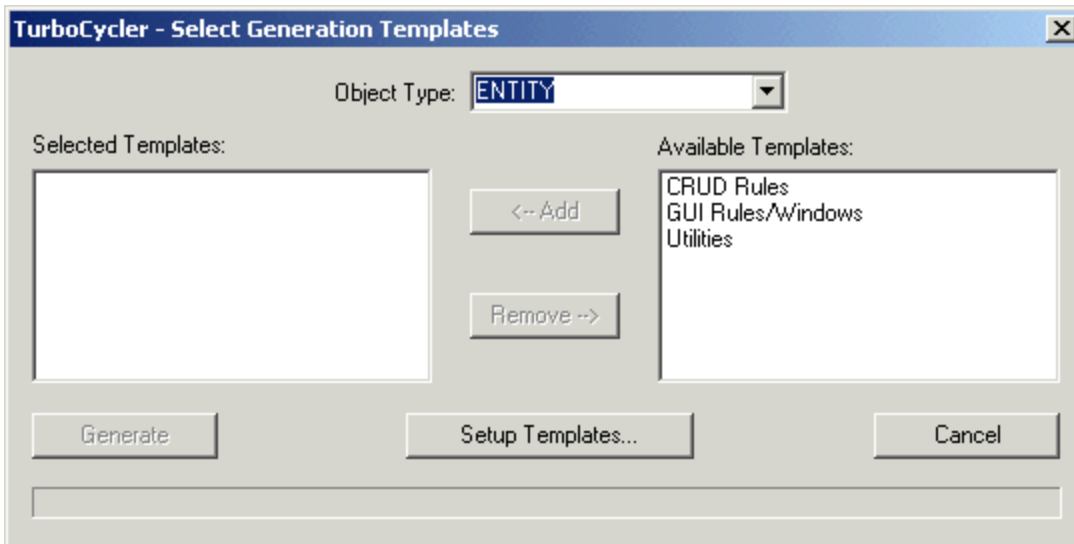
2. Look to see the system-generated name in the Screen Literal?Long field of the data views of your files in Hierarchy Diagram.
3. Edit each literal as necessary to change it permanently.
4. Select *File > Commit* to save session changes.

### Selecting TurboCycler Windows

This section assumes that you have created the Entity Relationship Diagram (ERD) and attributes described in [Defining an Entity Relationship Diagram](#) and have followed the rest of the procedures described up to this point. To select TurboCycler for Windows, complete the following steps:

1. Select the Entity Relationship Diagrammer window.
2. Select *TurboCycler* in the Analysis menu to display the TurboCycler window.
3. Make sure the templates selected in your window match [TurboCycler?Select Generation Templates window](#). Include all available templates (CRUD rules, GUI rules/windows, and utilities). Use the *Add* and *Remove* buttons to select the templates for generation.
4. When you have selected the templates, select the *Setup Templates* push button in the Select Generation Templates window.

### TurboCycler-Select Generation Templates window

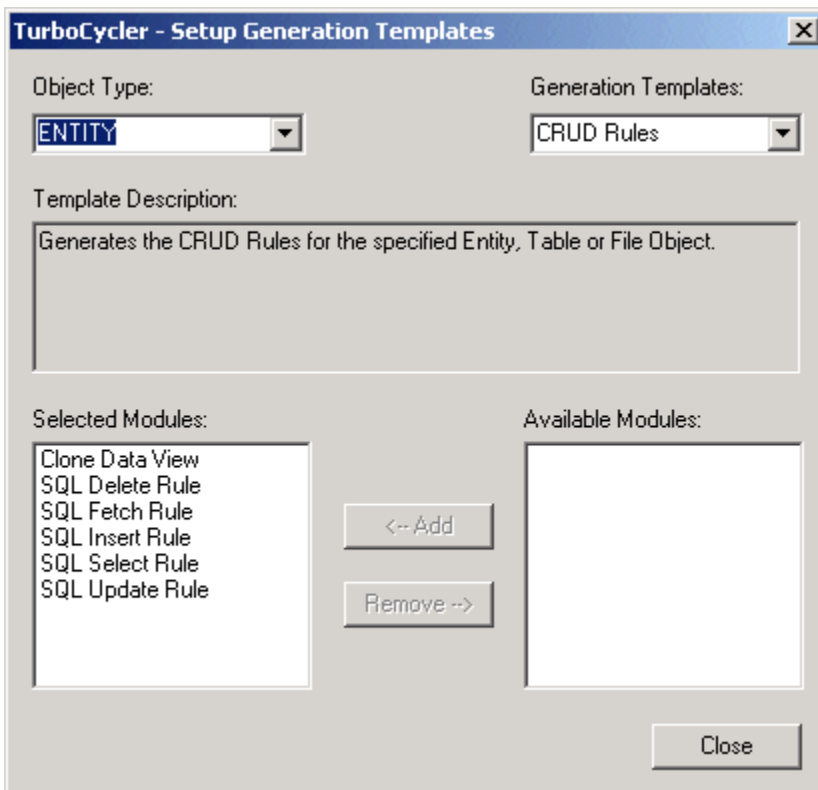


This displays the TurboCycler?Generation Template Setup window, which lists the modules available for each object type shown.

5. Select *Entity* in the Object Type list box and *CRUD Rules* in the Generation Template list box. Then select the modules for generation using the *Add* and *Remove* buttons.

6. Check all generation templates to make sure you select all the available modules for every template.

#### TurboCycler-Generation Template Setup window



7. Select *Close*.

8. If you have selected the appropriate entity templates and the modules that support it, select the *Generate* button to start generation in the TurboCycler dialog ([TurboCycler-Select Generation Templates window](#)).

The message line shows the status of the process. If there are errors, repeat the process.

9. Select *File > Commit* to save changes.



## Preparing the Application

After generating and committing your output, make additional modifications and programming changes and continue the process as usual with the preparation from the Construction Workbench. To prepare an application, complete the following steps in the Preparation Query window:

1. Select Generated files:
  - Delete the inappropriate tables.
  - Prepare files.
  - Create appropriate tables.
2. Select Generated Sets: prepare sets
3. Select Generated Windows: prepare windows
4. Select Generated Rules: prepare rules
5. Select Generated Functions: prepare functions

At this point, the preparation of the generated application is complete. Next, run the application (as described in [Procedure - Testing the Application](#)) to examine and test the output application, including the windows you prepared and painted in TurboCycler.

## Procedure - Testing the Application

Complete the following steps to test the application TurboCycler generated:

1. Select **Start > Programs > AppBuilder > Execution Clients > Windows Client**.
2. To execute from within the Construction Workbench, select **Run > Windows**.
3. Select **Agent Maintenance** in the menu.
4. The Execution Workbench asks if you want to start RuleView. Select **No**.
5. The Agent Detail Display window is displayed without any data.
6. Type test data in the displayed fields.
7. Select **File > Save** to save this agent data.
8. Type any name (for example, your own name) as additional test data in the Agent Detail Display window. Be sure to assign a different agent number. Save this data as you did in the previous step.
9. Select **Open** in the **File** menu to see the Agent List Display window.
10. Select the **Query** push button to display both agents.
11. Select **Query** and verify that both agent Smith and the other person's name appear in the Agent List Display window. (You can either query individual agents by their unique number or select the Query push button to see a list of all agents.)
12. To view the Agent Detail window for any agent, double-click the name of the agent. When you close the Agent Detail window, you return to the Agent Detail Display window.
13. To exit, select **File > Exit**.

# Using TurboCycler Developer Kit

## Using the TurboCycler Developer Kit

The TurboCycler Developer Kit provides an open architecture for automatically generating repository objects suitable for any platform and methodology. TurboCycler Developer Kit is a complementary product to the TurboCycler Standard Edition. With the Developer Kit, you define generation procedures using templates to automate steps in the software development lifecycle. You can write these generation procedures to meet your requirements.

To customize the template using the TurboCycler Developer's Kit, follow these general procedures:

- [Creating a Generation Template](#)
- [Editing and Compiling a Template](#)

## Creating a Generation Template

### Creating a Generation Template

TurboCycler generation templates are the procedures executed during a TurboCycler process. Use the TurboCycler template language to define a template and make generation decisions about what objects to generate in the repository. This involves the following:

- [Understanding the Generation Template](#)
- [Customizing the Generation Template](#)

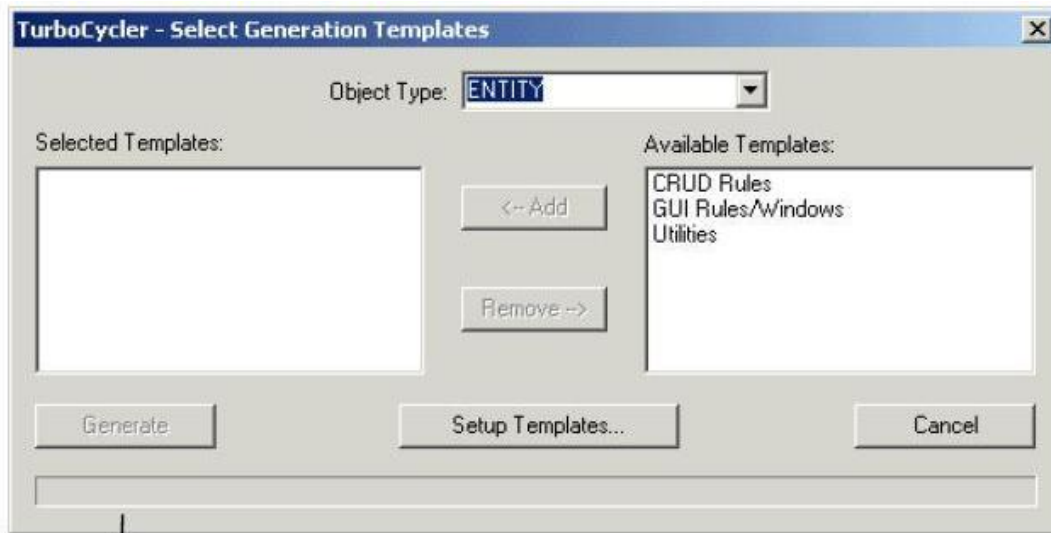
## Understanding the Generation Template

TurboCycler generation templates can access any object in the repository. For each object, the template can access all the editable properties of the object type. [TurboCycler Repository Types and Properties](#) discusses the complete list of object types and properties that TurboCycler supports in the Information Model.

A generation template must navigate through the repository to retrieve the information it needs to make generation decisions. For example, a template defined to generate an SQL Delete Rule for an AppBuilder file object needs information about the file, including information about the fields in the primary view. The template traverses the file hierarchy to obtain all the information it needs to generate the SQL Delete Rule.

A generation template can define object hierarchies, rules source code, and window panels, as shown in [Functionality of Generation Templates](#). You can generate a completely functional application by using these templates in the AppBuilder environment.

### Functionality of Generation Templates



Rule

```
*>-----<*>
*>- Rule: PRODUCT_SQL_DELETE -<*>
*>- SQL Delete Rule for the File: PRODUCT. -<*>
*>- Automatically Generated by AppBuilder*TurboCycler. -<*>
*>-----<*>
SQL ASIS
DELETE FROM PRODUCT
```

When building an object hierarchy, the generation template generates both the repository objects and the relationships between them and sets their properties. Templates can generate any type of object hierarchy including AppBuilder file, rule, and window hierarchies.

When defining rule source code, the generation template completely defines an AppBuilder rule. The source code associated with the rule can include any statement that the AppBuilder environment supports and can conform to your indentation and naming standards.

Window panels generated from a template can contain any type of control that the AppBuilder Window Painter supports. The template defines the position of the controls in the window and associates them with objects in the repository. [TurboCycler Window Controls](#) discusses the types of controls for window panels and their properties.

### Customizing the Generation Template

If an existing generation template does not generate objects, rules, or windows matching your requirements, you can create a new template or edit an existing template.

If you decide to create a new template, identify the repository information you need to make generation decisions and then identify those objects, rules, and windows that you are trying to generate. This information helps you design your new generation template efficiently.

If you change an existing template, you can change its object-naming standards, rules coding style and functionality, window presentation styles, and nearly anything else you want.



Default templates packaged with TurboCycler are not read-only or otherwise protected. Copy them before editing them and compile the template before running it.

# Editing and Compiling a Template

## Editing and Compiling a Template

Use the template editing environment to edit and compile templates. You can prepare templates by creating new ones or by changing existing ones. You must successfully compile templates before distributing them to end users or running them from the AppBuilder Construction Workbench. This involves the following:

- [Understanding the Environment](#)
- [Using the Command Line Compiler](#)
- [Using the Template Compiler](#)

In editing templates, the person who uses the results of your template programming is the *user*. The person who programs the output of your template is the *developer*.

## Understanding the Environment

The TurboCycler editing environment provides an interface for using your editor to edit template source files and two ways to compile templates. TurboCycler keeps all the templates in a single directory. You designate which directory to use. Source files for templates must have a file extension of .SRC and compiled templates have .TC.

TurboCycler Developer's Kit provides two compilers: a DOS command line compiler and a template compiler.

## Using the Command Line Compiler

The only function available at the command line is the ability to compile a source file (.SRC) to a template compile output file (.TC). The command line compiler has the name TCC.EXE, and its syntax is:

**TCC <Source> [<source>]**

where:

<source> = template source file name without the .SRC extension.

You specify only the file name because the .SRC file extension is assumed.

All other functions described in the following sections are available only through the template compiler.

## Using the Template Compiler

Use the template compiler to edit and compile templates interactively. It has a multi-threaded compiler that compiles templates in the background while you perform other work.

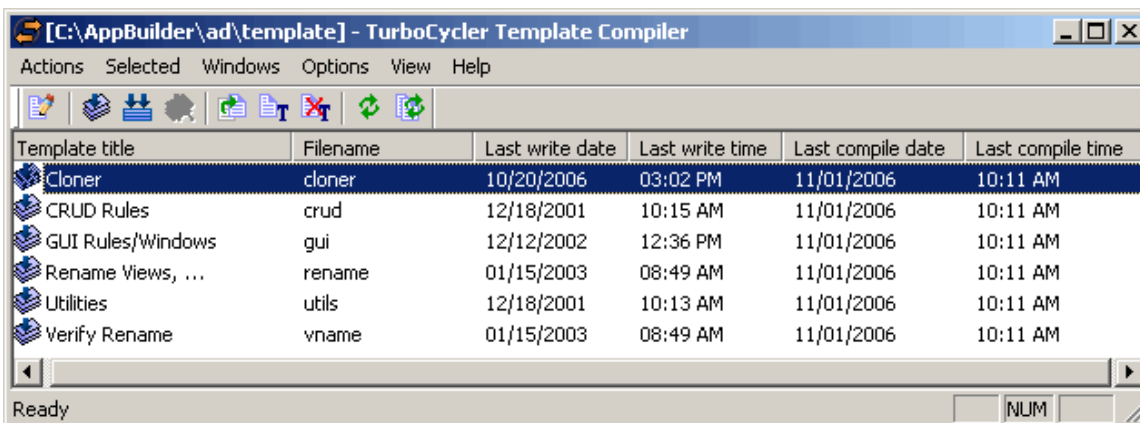
### [Procedure - Opening the template compiler](#)

To open the template compiler, complete the following steps:

**Select Start > All Programs > AppBuilder > TurboCycler Development Kit.**

The Template Compiler window opens ([TurboCycler Template Compiler window](#)).

### **TurboCycler Template Compiler window**



The main window for the template compiler contains a title bar identifying the template compiler window, a menu bar, a toolbar, and a work area with template information.

## Template Compiler Tools

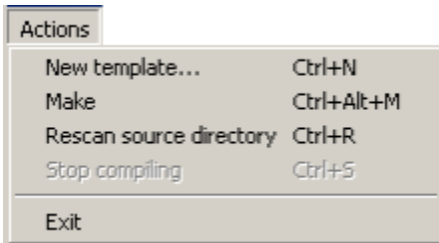
The following sections describe the menu options available for Template Compiler:

- [Actions Menu](#)
- [Selected Menu](#)
- [Windows Menu](#)
- [Options Menu](#)
- [View Menu](#)
- [Toolbar](#)

### Actions Menu

The menu bar contains an Actions menu, shown in [Template compiler Actions menu](#).

#### Template compiler Actions menu



[Actions Menu Choices](#) describes the Actions menu choices. Each Actions menu choice effects all listed templates, not just those currently selected.

#### Actions Menu Choices

Actions	Description
New Template	Displays the New Template window ( <a href="#">New Template window</a> ) that helps you prepare a new skeleton or outline template. You can define the file name, template title, and description; create hierarchy, rule, window, text, keywords, flat file, and component sections; and edit the template. You can also select this choice from the toolbar with the <i>New</i> push button.
Make	Compiles every template source file whose last write time is later than its last compile time. You can also select this choice from the toolbar with the <i>Make</i> push button.
Rescan source directory	Clears the list of templates shown and rereads the template source directory. You can also select this choice from the toolbar with the <i>Rescan</i> push button.
Stop compiling	Removes any templates waiting for compilation from the compile queue. Compiling stops when the current template completes. You can also select this choice from the toolbar with the <i>Stop</i> push button.
Exit	Exits the template compiler.

#### New Template window

**TurboCycler - New Template**

File name:

Template title:

Description:

Create sections

Hierarchy     Rule     Window

Text     Keywords     Flat file

Component

Options

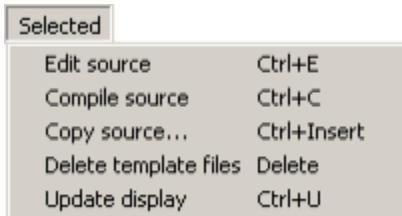
Edit template     Upper case

### Selected Menu

The menu bar contains a Selected menu, shown in [Template Compiler Selected menu](#).

### Template Compiler Selected menu



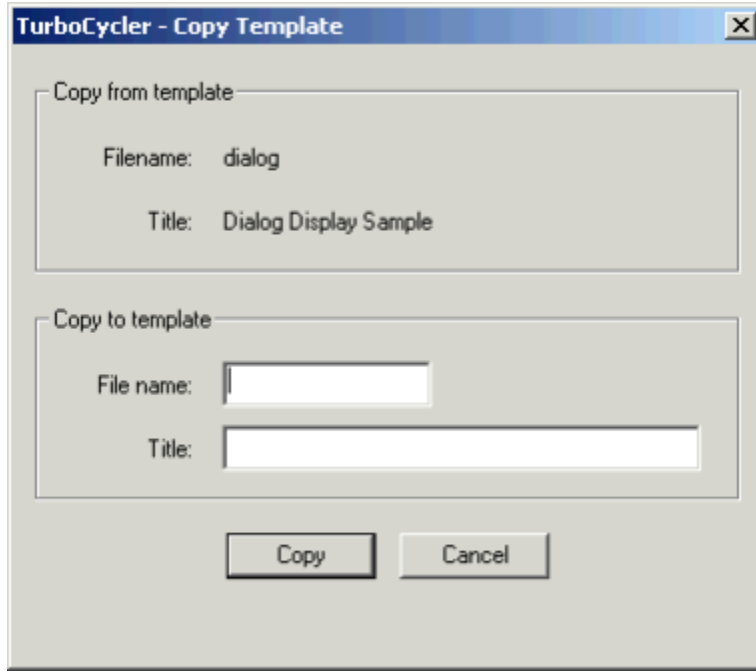
[Selected Menu Choices](#) describes the Selected menu items and descriptions. These menu choices affect only the selected templates.

### Selected Menu Choices

Menu Item	Description
Edit source	Opens the editor for the selected templates. (Depending on options set in the Set Editor window, you can open the editor once for each selected template or once for all selected templates.) You can also select this choice from the toolbar with the <i>Edit</i> push button.
Compile source	Queues the selected templates and compiles them in the order they appear in the window. You can also select this choice from the toolbar with the <i>Compile</i> push button.
Copy source	Opens the Copy Template window ( <a href="#">Copy Template window</a> ), which enables you to copy each of the selected templates and to give the copy a name and title. You can also select this choice from the toolbar with the <i>Copy</i> button.

Delete template files	Deletes the source and compiled files for all selected templates. You can also select this choice from the toolbar with the <i>Delete</i> button.
Update display	Rescans the source files of the selected templates to check whether the title or last-written times have changed. (Use this when you use the editor to make changes to a template.) You can also select this choice from the toolbar with the <i>Update</i> button.

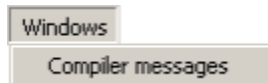
### Copy Template window



### Windows Menu

The menu bar contains a Windows menu, shown in [Template Compiler Windows Menu](#).

#### Template Compiler Windows Menu

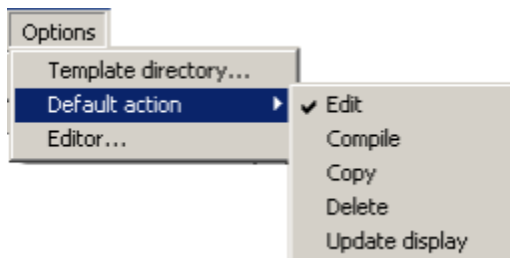


The Compiler messages choice displays a window that shows template compilation and error messages.

### Options Menu

The menu bar contains an Options menu, shown in [Template Compiler Options menu](#).

#### Template Compiler Options menu

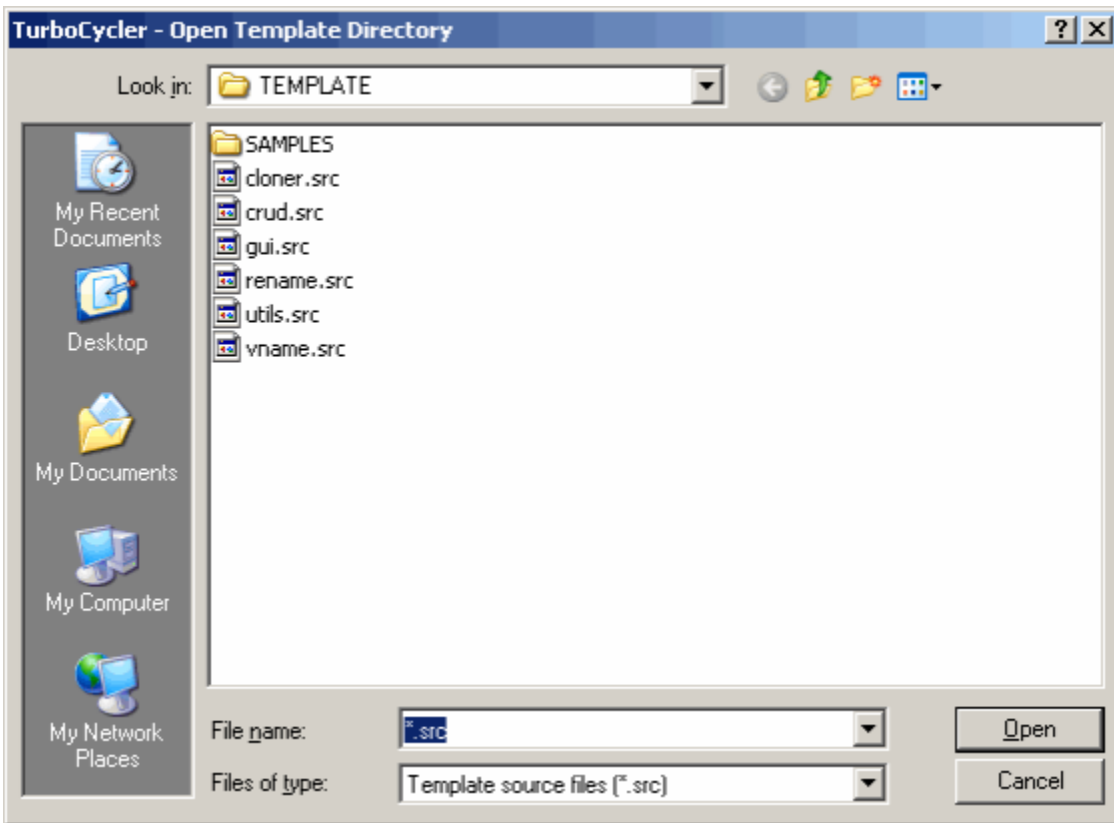


[Options Menu](#) describes the Options menu choices and describes their behaviors.

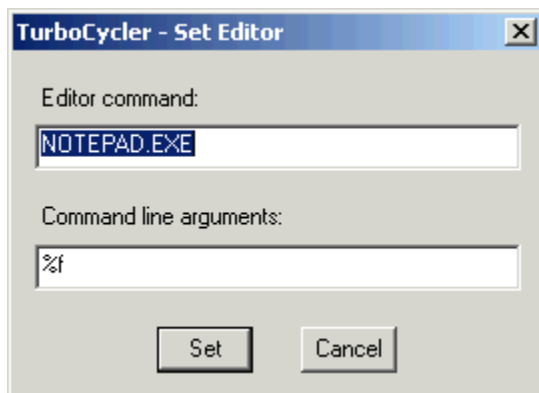
### Options Menu

Menu Items	Description
Template directory	Displays the Open Template Directory window ( <a href="#">Open Template Directory window</a> ) that specifies the directory where templates reside on the developer's system.
Default action	Specifies the action to be performed when you double-click a template or press <i>Enter</i> . The choices are Edit, Compile, Copy, Delete, and Update display.
Editor	Opens the Set Editor window ( <a href="#">Set Editor window</a> ) in which you can specify your editor. The editor command field defines the executable path name, and the editor command line field specifies the command line arguments to pass to the editor. If your editor supports passing multiple file names on the command line, use the %a specifier. Otherwise, use the %f specifier to open the editor separately for each selected template. You can also specify any other options to pass to the editor.

**Open Template Directory window**



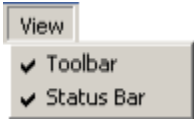
**Set Editor window**



**View Menu**

Use the View menu to choose whether to display the toolbar and the status bar while you are working in the template compiler window. To display either bar, click the name of the bar in the drop-down list. A check mark appears. To hide either bar, click again to remove the check mark.

**Template Compiler View menu**



**Toolbar**









For fast access to the Actions and Selected menu choices, use the toolbar at the bottom of the template compiler window ([Template Compiler toolbar](#)).

**Template Compiler toolbar**




The toolbar presents the menu choices as push buttons ([toolbar Push Buttons](#)).

**toolbar Push Buttons**

Icon	Push button	Menu choice equivalent
	Edit	Edit source choice in Selected menu
	Compile	Compile source choice in Selected menu
	Make	Make choice in Actions menu
	Stop	Stop compiling choice in Actions menu
	Copy source	Copy source choice in Selected menu
	New template	New template choice in Actions menu
	Delete template files	Delete template files choice in Selected menu
	Update display	Update display choice in Selected menu







	Rescan source directory	Rescan source directory choice in Actions menu
---	-------------------------	--

## Template Compiler Work Area

The work area has a row for every template in the template directory. [Template Compiler Work Area](#) lists the information for each template.

### Template Compiler Work Area

Dialog columns	Descriptions	
Status icons	Four icons indicate template status. Each template displays one icon indicating its status:	
		Source
		Compiled
		Compilation in progress
		Queued for compilation
Template title	The assigned title of the template.	
Filename	The file name of the template as stored in the designated directory. (Preassigned file extensions are .SRC for a source template and .TC for a compiled template.)	
Last write date	The date of the last template writing.	
Last write time	The time of the last template writing.	
Last compile date	The date of the last successful template compilation.	
Last compile time	The time of the last successful template compilation.	

## TurboCycler Template Language

### TurboCycler Template Language

The TurboCycler template language defines how to generate repository objects from other repository objects. The logic of a template can include standard language statements (such as IF?ELSE and WHILE) and more specific TurboCycler statements. TurboCycler statements support repository navigation, queries, and generation.

This section presents the important high-level concepts for developing templates, followed by the rudimentary statements. If you are not familiar with basic programming techniques, you might need to review those sections in the supporting statements and structures section before learning the template sections and concepts.

You can find detailed information about language usage, syntax, and function of the AppBuilder Rules Language in the *Rules Language Reference Guide*.

## Flow Diagrams Overview

Flow Diagrams define the syntax of the template language. The syntax is shown in flow diagrams that show how the parts of the template language relate and how to use each statement.

- [Using Conventions and Symbols](#)
- [Reading a Diagram](#)
- [Inserting Multiple-line and Single-line Comments](#)



Flow diagrams might not illustrate every condition and restriction of each statement. Read the description of each statement for complete information.

### Using Conventions and Symbols

The following conventions and symbols appear in the syntax flow diagrams:

- A WORD in all capital letters is a template language keyword.
- A word not capitalized or italicized is defined in another flow diagram.
- A "word" in quotations is a value from outside the template language that you must provide when you code the statement, such as the name of a field or a string literal.

#### *Symbols Used in Syntax Flow Diagrams*

Symbol	Description
	Flow of statement starts
	Flow continues on next line or may include other path
	Flow continued from previous line
	Flow can branch in either direction
	Flow of statement ends

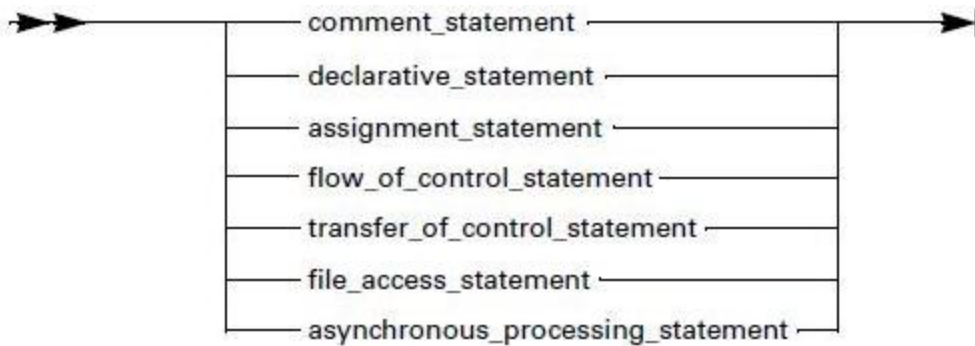
### Reading a Diagram

Complete the following steps to understand the syntax of a diagram:

1. Start at the double-headed arrow on the left side and go to the end of the diagram.
2. Follow any one of the possible line paths from left to right. Any path that you can traverse from left to right results in valid syntax. In whichever line you follow, you must use all words or symbols that you move through on that line.
3. You cannot go back to the left unless there is a loop, which is indicated by an arrow on its left end and appears above another line. You can go around a loop any number of times.

#### Statement Sample

This sample shows a simple flow diagram. It illustrates the basic approach that the TurboCycler flows follow:



## Inserting Multiple-line and Single-line Comments

To enter comments, use the // token that ends with a new line character.



Both the // token and the new line are discarded from the input. For example:

```

USE FIELD(x)
// Get the name of the field
SET Y = QUERY NAME OF X
ENDUSE
  
```

## Template Language Statements

This section contains the flow diagrams for the TurboCycler template language statements. It has two parts. The first part describes the template statements, and the second part describes supporting statements.

Follow the flows as described above and read the explanations for details and programming techniques.

The template statements contain the language flows for the following:

- [TEMPLATE Statement](#)
- [USAGES Block](#)
- [HIERARCHY Block](#)
- [RULE Block](#)
- [WINDOW Block](#)

There are also [Other Blocks](#) and [Supporting Statements and Expressions](#).

### TEMPLATE Statement

The TEMPLATE statement is the starting point of a TurboCycler template. It provides the template with a title and description, by which the user identifies the template during the TurboCycler Standard Edition generation process.

A template contains only one TEMPLATE statement and has a unique name. A template also contains a usages block that determines the initiation process for this template. A template can contain multiple hierarchy, rule, and window blocks grouped in modules by a common module name.

#### "template\_name"

A string constant that identifies this template. This is the title of the template that the user sees during a TurboCycler Standard Edition generation. For example, the CRUD Rules default template is named "CRUD Rules."

#### DESCRIPTION?ENDDescription

Keywords that surround the "template description."

#### "template\_description"

A string constant that describes this template. This is the only description for a template that the user sees during a TurboCycler generation. For example, the default template named CRUD Rules has the description "Generates the CRUD Rules for the specified Entity, Table or File object." The description can be as short or long as you want; it may include multiple lines.

### **usages\_block**

A template language statement that provides the entry point that TurboCycler uses to run the template. In most cases, TurboCycler starts with a single repository object and proceeds to extract information about that object needed to generate hierarchies, rules, or windows. Entry points for the default templates are entity, a many-to-many relationship, table, and file.

A usages block takes a single object and performs assorted repository queries. It sets up variables on which other blocks depend. Refer to [USAGES Block](#) for details.

### **hierarchy\_block**

A template language statement that defines a hierarchy for any object type, such as rule hierarchy, window hierarchy, file hierarchy, and other hierarchies. The hierarchy block creates objects, sets their properties, creates relationships between objects, and sets properties for the relationships. Refer to [HIERARCHY Block](#) for details.

### **rule\_block**

A template language statement that defines the source code or rules source for a specific rule. For example, the SQL Delete Rule in the CRUD Rules Template. Refer to [RULE Block](#) for details.

### **window\_block**

A template language statement that paints a window panel for a specific window; for example, the Detail Display Window in the GUI Rules/Window Template. The window block places controls on the window and establishes links to appropriate repository objects. Refer to [WINDOW Block](#) for details.

### ***Special Notes***

Different blocks can be strongly dependent on each other. For example, generated rule source depends on a rule hierarchy. You can force multiple blocks to be generated by giving them all the same module name, thus reinforcing their dependence.

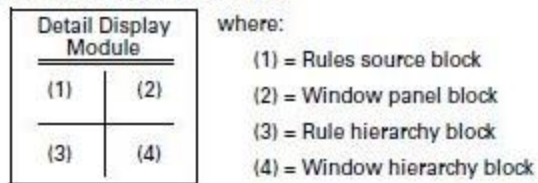
Another example of block interdependence is a rule conversing a window. Using a template to generate a rule conversing a window requires four heavily dependent blocks:

- Rule source block
- Window panel block
- Rule hierarchy block
- Window hierarchy block

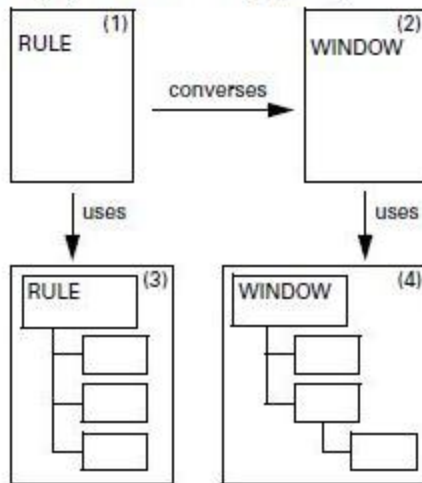
Reinforce this dependence by giving all the same module name. [Example of mutually dependent module blocks](#) shows this relationship.

### ***Example of mutually dependent module blocks***

The Detail Display Module is composed of:

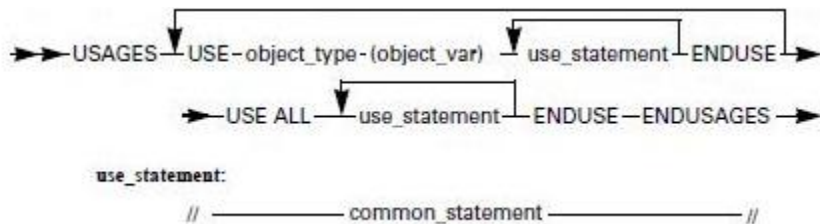


The Detail Display Module illustrated graphically:



## USAGES Block

The usages block accepts specified repository object types as valid entry points into the generation process that the template describes. It associates the current template with repository objects of the object types you specify. Control which objects are selected for generation with the USAGES statement. TurboCycler generates output objects only for the object types you specify in the usages block.



Thus, your selection influences the tools from which you can access this template. For example, if you define a template to accept entities as input, the template is available from only those tools supporting entities, such as Entity Relationship Diagrammer, Hierarchy Diagrammer, and Matrix Builder.

### USAGES...ENDUSAGES

Keywords that surround the usages block.

### USE...ENDUSE

Keywords that surround your specification of a valid object type, object variable, and use\_statement. Each USE...ENDUSE set supports only one "object\_type" specification. Repeat this set of keywords for each object type that you want the template to support.

### object\_type

The object type from which you want this template to support generation. Any object type in the Information Model is valid, such as RULE, WINDOW, VIEW, or FIELD. You can select more than one object type by using multiple USE?ENDUSE repetitions, as shown in the flow diagram. Refer to [Object Types and Properties](#) for a list of object types.

The object type can also be an open drawing. The USE type supports processing the symbols on an open drawing through a template. Refer to [Processing Symbols on an Open Drawing](#) for details.

### (object\_var)

An object variable that stores a reference to the repository object passed into the template from an AppBuilder development tool.

#### use\_statement

A sequence of statements that are part of the usages block. These statements insert the common statement services of use-dependent queries and variable initializations such as RETURN, DEBUG, IF, FOR, WHILE, and SET. See [common\\_statement](#).

#### USE ALL...ENDUSE

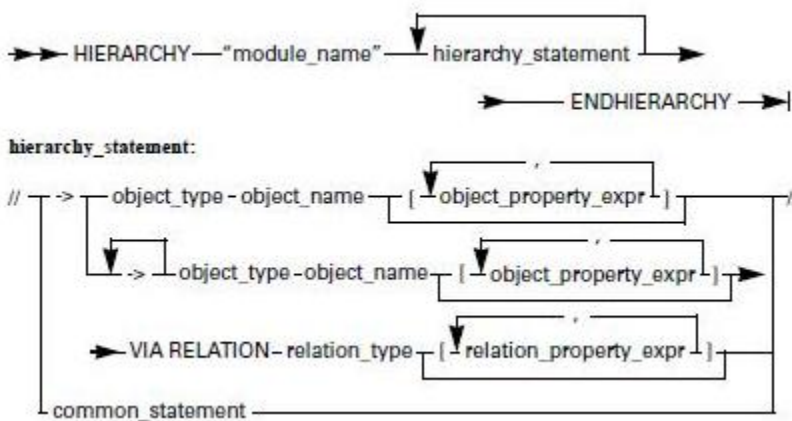
Keywords that surround your specification of all shared use statements in a usages block. The use statements contained in the USE ALL?ENDUSE set are applied to each USE?ENDUSE set. Use this statement to code any function that you want applied to all USE?ENDUSE invocations only once. The generation process then executes the complete set for each USE?ENDUSE set defined in the usages block.

#### common\_statement

A sequence of statements common to several block statements including this one. The common statement provides common services for use-dependent queries and variable initializations such as RETURN, DEBUG, IF, FOR, WHILE, and SET. Refer to [Common Statement](#) for details.

### HIERARCHY Block

The hierarchy block defines a hierarchy for any object type such as a rule hierarchy, window hierarchy, or file hierarchy. The hierarchy block creates objects and sets properties for them, creates relationships between objects, and sets properties for those relationships.



The hierarchy block that TurboCycler generates closely resembles the hierarchy that AppBuilder prepares in both content and form in the following respects:

- Create repository objects
- Set properties for those objects
- Create relationships between objects
- Set properties of the relationships

#### Double Arrow in Hierarchy Block

If the first arrow statement in a hierarchy block consists of a single arrow and the object named already exists in the repository, TurboCycler deletes all existing relations to child objects. This feature allows the template to control precisely what objects the hierarchy contains. This feature can be undesirable, however, if the template is meant to update an existing object without disturbing existing child relations. In this case, you should begin the hierarchy with two arrows instead of one.

After executing the following block, the rule named TEST is guaranteed to have only the view VIEW1 as a child:

```
HIERARCHY "Example"  
RULE "TEST"  
VIEW "VIEW1" VIA RELATION OWNS_VIEW  
ENDHIERARCHY
```

Alternatively, the following block simply appends the view VIEW1 as a child of the rule TEST without disturbing any existing children.

```

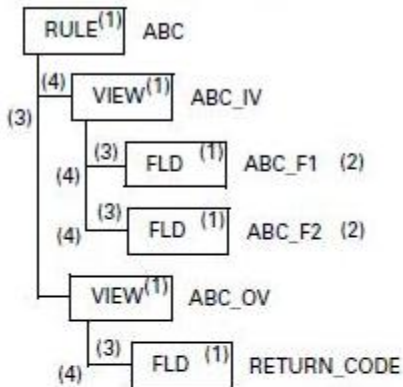
HIERARCHY "Example"
RULE "TEST"
VIEW "VIEW1" VIA RELATION OWNS_VIEW
ENDHIERARCHY

```

[Comparison of AppBuilder and TurboCycler hierarchies](#) compares a hierarchy diagram and TurboCycler, and shows their similarities.

### Comparison of AppBuilder and TurboCycler hierarchies

Sample of a hierarchy diagram



Essential elements of a hierarchy

- (1) Create object
- (2) Set property
- (3) Create relationship
- (4) Set properties of relations

Sample of a hierarchy block, with defined elements

```

-> RULE "ABC"
-> -> VIEW "ABC_IV"
                                VIA RELATION OWNS_VIEW [SEQUENCE=10,
USAGE="INPUT"]
-> -> -> FIELD "ABC_F1" [TYPE="INTEGER", LENGTH=15]
                                VIA RELATION VIEW_INCLUDES
[SEQUENCE=10]
-> -> -> FIELD "ABC_F2" [TYPE="INTEGER", LENGTH=15]
                                VIA RELATION VIEW_INCLUDES
[SEQUENCE=20]

```

### HIERARCHY...ENDHIERARCHY

Keywords that surround the hierarchy block.

#### "module\_name"

A string constant that provides the name for the module to be generated. The name must be a unique phrase that accurately distinguishes this module from any other. The user of a template selects the module for generation using the TurboCycler Setup Generation Templates window.

#### hierarchy\_statement

A sequence of statements that are part of the hierarchy block. These statements insert the unique hierarchical statements for creating entities and templates. The hierarchy\_statement also inserts the common statement services of hierarchy-dependent queries and variable initializations such as RETURN, DEBUG, IF, FOR, WHILE, and SET. Refer to [common\\_statement](#).

->

A hierarchy symbol that establishes the generational level for the following definition of an object.

#### object\_type

A parameter that defines the type of object being generated. Refer to [Object Types and Properties](#) for a list of object types.

#### object\_name

Either a string expression or an object expression that specifies the name of the object being generated.

[...]

A set of brackets encloses properties of the object that are to be set.

Use a comma to separate each object property expression (see [object\\_property\\_expr](#) below).

#### **object\_property\_expr**

An object property expression that sets the property of the object to the specified value. Refer to [Object Types and Properties](#) for a detailed list. The format for this statement is PROPERTY = string\_expr or PROPERTY = int\_expr.

#### **VIA RELATION**

A keyword that creates the relationship for the object and specifies how the child object is to be related to the parent object.

#### **relation\_type**

A parameter that defines the type of relationship for the object. Refer to [Relationship Types and Properties](#) for a detailed list.

#### **relation\_property\_expr**

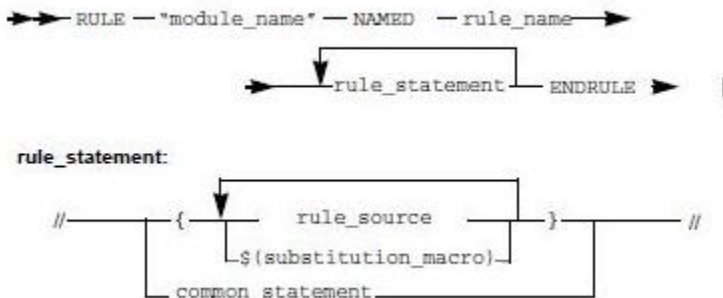
A parameter that sets the properties of the relationship to the specified value. Refer to [Relationship Types and Properties](#) for a detailed list. The format for this statement is PROPERTY = string\_expr or PROPERTY = int\_expr.

#### **common\_statement**

A sequence of statements common to several block statements including this one. The common statement provides common services for hierarchy-dependent queries and variable initializations such as RETURN, DEBUG, IF, FOR, WHILE, and SET. Refer to [Common Statement](#) for details.

### **RULE Block**

The rule block defines the rules source code for a specific rule. An example of a rule block is the SQL Delete Rule in the CRUD Rules Template. If the repository already has existing source code for a specified rule, any code you generate with an identical name overwrites the existing code.



The rule block creates rule source code as well as all common template statements for manipulating variables, looping and branching, returns, and debugging. You can recognize rule source by the curly braces that enclose it. Everything you write within the curly braces goes into the rule code, either directly as rule source code or indirectly by macro substitution.

#### **RULE...ENDRULE**

Keywords that surround the rule block.

#### **"module\_name"**

A string constant that provides the name for the module to be generated. The name must be a unique phrase that accurately distinguishes this module from any other. Select the module for generation using the TurboCycler Setup Generation Templates window.

#### **NAMED**

Keyword that specifies the rule that the "rule\_name" parameter identifies.

#### **rule\_name**



Either a string expression or object expression that identifies the rule to be generated in the repository.

#### **rule\_statement**

A sequence of statements that are part of the rule block. These statements include the unique "curly braces" rules code and the common statement services of rule-dependent queries and variable initializations such as RETURN, DEBUG, IF, FOR, WHILE, and SET. Refer to [common\\_statement](#) below.

{...}

Curly brace symbols indicate that the enclosed statements are rules code. Rules code can be a single statement or multiple statements. The source of rules code is either direct as a copy from the template (see [rule\\_source](#)) or indirect from a substitution macro (see [\\$\(substitution\\_macro\)](#)).

#### **rule\_source**

Everything that is enclosed in braces, except the `$(substitution_macro)` format, is put directly into rules code. Code the Rules Language statement that you want incorporated. See [Special Notes](#) for more information.

#### **\$(substitution\_macro)**

Everything that is enclosed in the "\$..." format becomes rules source code by macro substitution. Use this statement to place the value of template variables or expressions into the rules source. See [Special Notes](#) below for more information.

#### **common\_statement**

A sequence of statements common to several block statements including this one. The common statement provides common services for rule-dependent queries and variable initializations such as RETURN, DEBUG, IF, FOR, WHILE, and SET. Refer to [Common Statement](#).

#### **Special Notes**

The RULE and NAMED keywords identify the name of the module for the TurboCycler user and rule object to be added to the repository. For example, the statement:

```
RULE "SAMPLE" NAMED "B"
```

creates a module called SAMPLE, which generates rules source code for object B in the repository. If source code exists for object B, the new rule source overwrites the existing code.

If a template has more than 72 characters of rules source per line, TurboCycler breaks each line at the last blank space before the limit, making as many lines as necessary to contain the code.

To create rules source code, code inside the curly braces, including any macro substitutions. The position of curly braces determines the indentation of the rules source code, as shown in the following example. (Refer to [Template Samples](#) for a detailed sample of a real rule source generation.)

```
RULE "Sample" NAMED "B"
SET V = GET VIEW NAMED "V1"
\{ *> Rule generated by TurboCycler <*)
\}
SET FROM = 10
\{ DCL
I INTEGER;
ENDCL
MAP $(FROM) TO I
MAP $(FROM + 20)
TO RETURN_CODE OF $(QUERY NAME OF V)
\}
ENDRULE
```

TurboCycler generates the following rules source code:

```

*> Rule generated by TurboCycler <*
DCL
I INTEGER;
ENDCL
MAP 10 TO I
MAP 30
TO RETURN_CODE OF V1

```

### Object Property Comparisons

All comparisons to object properties in the retrieval clause are case insensitive. For example, the following statement retrieves the VIEW object named V1:

```
SET V = GET VIEW NAMED "V1"
```

Properties are not converted to upper case during reads and writes.

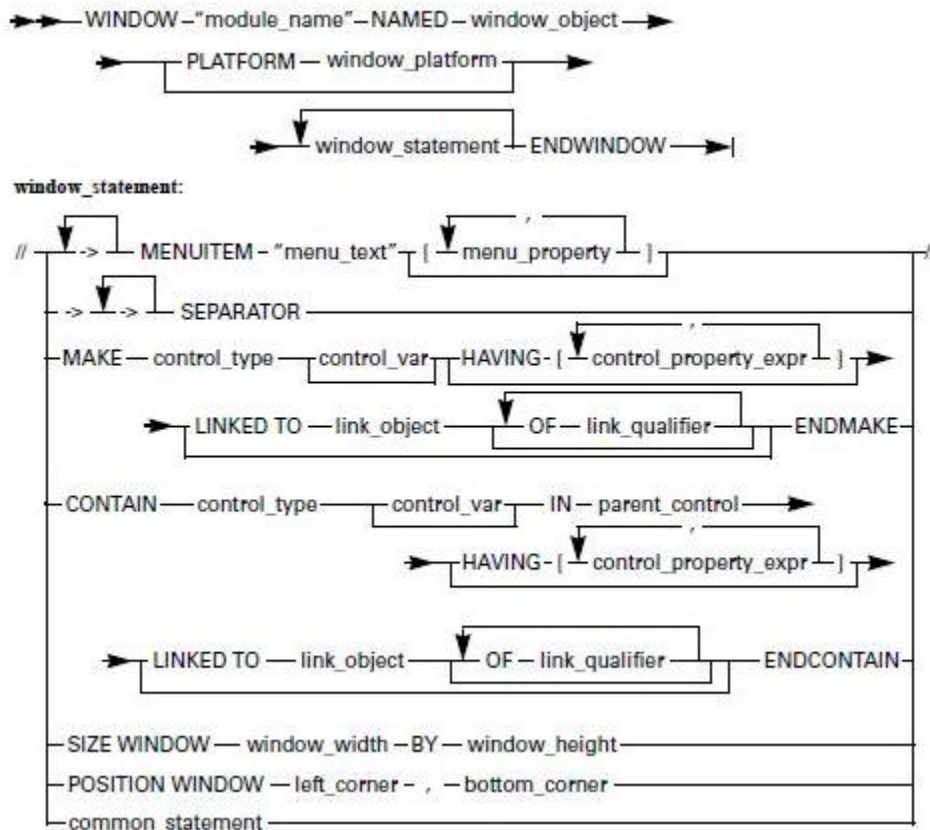
### Property Value Expression Types

Property values are converted to the same expression type as the query value in HAVING and WITH clauses. For example, the following expression works as expected.

```
SET X = CHILD FIELD OF VIEW HAVING \[LENGTH = 33\]
```

## WINDOW Block

The window block paints a window panel for a specific window, such as the Detail Display Window in the GUI Rules/Window Template. The window block places controls on the window and links them to appropriate repository objects.



Building menus in a window is similar to making menu hierarchies with the Window Painter menu editor. The statements in a window block specify window panels in a way that corresponds to the Window Painter graphic interface. For example, the window block MAKE statement corresponds to dropping controls in Window Painter. However, TurboCycler does not support creating or updating cascading menus from the

template.

The PLATFORM keyword and its parameter window\_platform specify the GUI target of the window. The default value is PWS\_GENERIC. You can create 3270 or other types of window panels, but you must observe documented AppBuilder restrictions on those panels.

The following example shows how to code a window block statement that constructs a menu. The example creates a menu for the File item, consisting of lines for New, Save, and Exit, each of which has a separator between them.

```
REM --- Generate a sample window \---;
WINDOW "Sample Window" NAMED MyDetailSample
SET MyCDV = GET VIEW NAMED MyClonedDataView
REM --- Make the Detail Sample Display \---;
\-> MENUITEM "&File"
\-> \-> MENUITEM "&New" \[HPSID="New"\]
\-> \-> SEPARATOR
\-> \-> MENUITEM "&Save" \[HPSID="Save", GREYED="TRUE"\]
\-> \-> SEPARATOR
\-> \-> MENUITEM "&Exit" \[HPSID="Exit"\]
ENDWINDOW
```

See [Template Samples](#) for detailed coding of template windows.

## WINDOW...ENDWINDOW

Keywords that surround the window block.

### "module\_name"

A string constant that provides the name for the module to be generated. The name must be a unique phrase that accurately distinguishes this module from any other. The user of a template selects the module for generation using the TurboCycler Setup Generation Templates window.

### NAMED

Keyword that specifies the window that the "window\_name" parameter identifies.

### window\_name

Either a string expression or object expression that names the window to be generated in the repository.

### window\_statement

A sequence of statements that are part of the window block. These statements include the window block-specific statements as well as the common statement services of window-dependent queries and variable initializations such as RETURN, DEBUG, IF, FOR, WHILE, and SET. Refer to [scommon\\_statement](#).

->

A menu hierarchy symbol that establishes the generational level for the menu item entry.

### MENUITEM

A keyword that starts the text for a menu line and assigns its properties.

### "menu\_text"

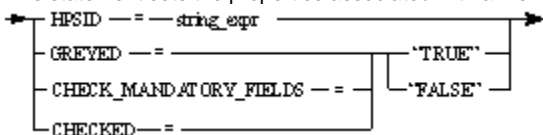
The menu text (a string expression) that provides the text of a menu line.

[...]

A set of brackets that enclose properties of the menu item to be set.

### menu\_property

This statement sets the properties associated with a menu item. Refer to the following flow diagram for the parameters of this variable.



## **SEPARATOR**

A keyword that creates a horizontal divider in a list of menu lines.

## **MAKE...ENDMAKE**

A set of keywords that makes a control on the window. You can also define the properties of the control and link it to a repository object in the MAKE...ENDMAKE statement.

### **control\_type**

A parameter that describes the kind of window control being generated. Refer to [Window Statement Control Types](#) for details of control types.

### **control\_var**

A control variable that stores a reference to a window control and gives access to the control later; essentially it names the object being created by the MAKE. Use it with functions like LEFTOF or BOTTOMOF.

## **HAVING**

A keyword that sets the properties for the window controls. Each control type has its own properties defined by the parameter following "control\_property\_expr".

### **control\_property\_expr**

A parameter that defines the properties for the window control. Refer to [Window Control Properties](#) for the detailed parameter list of the properties of window controls. The format of this expression is PROPERTY = string or PROPERTY = integer.

## **LINKED TO**

A keyword that connects the control being created to an existing repository object. You can link controls only to objects that are descendents of a window view.

### **link\_object**

An object expression that names the repository object associated with the made control object being created.

## **OF**

A keyword required when the link to a control is a descendent that is not a direct child of the window view. Each level needed to link to a direct child of the window view requires an OF keyword. Use the OF statement multiple times to link to the appropriate level of the window view.

### **link\_qualifier**

An object expression that identifies the descendent of the window view for which the link is being made.

## **CONTAIN?ENDCONTAIN**

A set of keywords that also define the type, name, and properties of a window control object. The CONTAIN statement makes a window control within a previously defined control. Use CONTAIN to make CELLS in a SPREADSHEET or to define the axes of a CHART. The format of these parameters is the same as the MAKE keywords.

## **IN**

A keyword that identifies the parent control of the contained control.

### **parent\_control**

A control variable that names the parent control object of the contained control being defined.

## **SIZE WINDOW**

A keyword that defines the dimensions of the window being generated.

### **window\_width**

An integer expression giving the width of the window in pixels.

## **BY**

A keyword that indicates another window dimension follows.

**window\_height**

An integer expression giving the height of the window in pixels.

**POSITION WINDOW**

A keyword that defines the location of the window being generated.

**left\_corner**

An integer expression that locates the left side of the window being generated.

**bottom\_corner**

An integer expression that locates the bottom of the window being generated.

**scommon\_statement**

A sequence of statements common to several block statements including this one. The common statement provides common services for window-dependent queries and variable initializations such as RETURN, DEBUG, IF, FOR, WHILE, and SET. Refer to [Common Statement](#) for details.

## Other Blocks

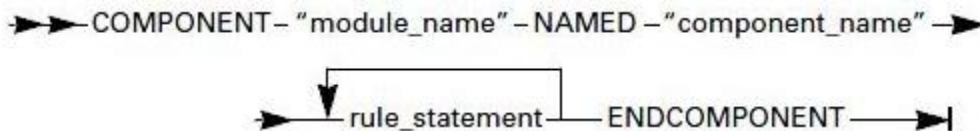
These include:

- [COMPONENT Block](#)
- [FLATFILE Block](#)
- [KEYWORDS Block](#)
- [TEXT Block](#)
- [USE ANY Block](#)
- [DIALOG Block](#)

See also the [Supporting Statements and Expressions](#).

### COMPONENT Block

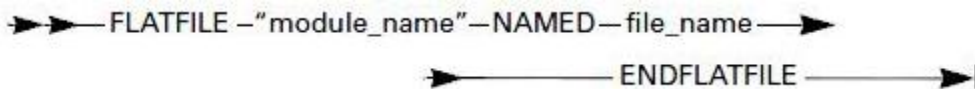
This block is available for composing the source of a COMPONENT. Its syntax is:



The statements supported are the same as for the RULE block statement. For details, refer to the [RULE Block](#).

### FLATFILE Block

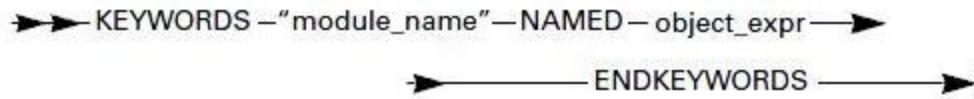
This block is available for composing text files. Its syntax is:



The statements allowed are the same as for the RULE block statement. For details, refer to the [RULE Block](#).

### KEYWORDS Block

This block is available for composing the KEYWORDS of an object. Its syntax is:



The object\_expr translates to the repository object for which the KEYWORDS are to be written. The statements allowed are the same as for the RULE block statement. For details, refer to the [RULE Block](#).

### TEXT Block

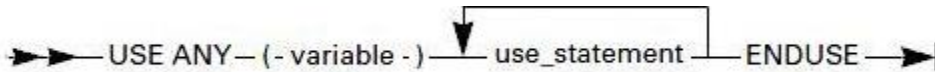
A new block is available for composing the TEXT of an object. Its syntax is:



The object\_expr translates to the repository object for which you are writing the TEXT. The statements allowed are the same as for the RULE block statement. For details, refer to the [RULE Block](#).

### USE ANY Block

This type of USE block accepts any object type for processing. The syntax of the USE ANY statement is:



If you use the USE ANY?ENDUSE block, put it after all your USE?ENDUSE blocks and before the USE ALL?ENDUSE block.

### DIALOG Block

A block type is available for composing user dialogs. The block is:

**DIALOG "module\_name"**

// Dialog statements  
ENDDIALOG

Use the DIALOG block to prepare and display a dialog box to the user while the template is running. This can be used to ask questions of the user and to display the results of the generation process.

#### **Specifying Size and Position of User Dialogs and Controls**

You can specify the size and position of user dialogs and controls by specifying their coordinates in dialog units. A dialog unit is defined as 1/8 of the average height of the system font and 1/4 the average width of the system font. In other words, there are about 4 dialog units for every character width you specify, and 8 dialog units for every character height. For example, if you wanted to create a dialog that was 40 characters in width and 50 characters in height, you would make the width 160 units and the height 400 units.

#### **Creating Controls on a Dialog**

Create controls on a dialog using the MAKE...ENDMAKE statement as used in the WINDOW block. All of the controls can be assigned to a variable, and any of the properties can be queried using the QUERY statement. Some of the controls can have a LINKED TO clause, which is always optional. Unlike the WINDOW block MAKE statement, the LINKED TO clause must always specify a template variable.

You can use the HAVING clause of the MAKE statement to set the properties of dialog controls. You can also set these properties using the new SET PROPERTY OF VARIABLE function.

#### **Setting Properties of Dialog Controls**

The following table lists the properties of the dialog controls and describes the characteristics of those properties. An N in the table means Microsoft Windows.

#### **Dialog Control Properties**

	A	B	B	C	C	D	G	H	H	H	L	M	M	P	R	T	V	W	W
	U	O	O	O	T	E	R	E	O	P	E	A	U	A	E	A	E	I	O
	T	R	T	N	L	F	O	I	R	S	F	R	L	S	A	B	R	D	R
	O	D	T	T	3	A	U	G	Z	I	T	G	T	S	D	S	T	H	D
	S	E	O	O	D	U	P	H	S	C		I	N	W	O	T	S		W
	C	R	M	L		L	S	T	C					O	N	O	C		R
	R				T	T	A	R	O					R	L	P	R		A
	O																		P
	L																		
	L																		
CHECKBOX			N	N			N	N		N	N					N		N	
EDIT_FIELD	N	N	N	N	N		N	N	N	N	N	N		N	N	N		N	
GROUPBOX			N	N			N	N		N	N					N		N	
LISTBOX			N	N	N		N	N	N	N	N		N			N	N	N	
MULTILINE_EDIT	N	N	N	N	N		N	N	N	N	N				N	N	N	N	N
PROTECTED_EDIT_FIELD		N			N														
PUSH_BUTTON			N	N		N	N	N		N	N					N		N	
RADIO_BUTTON			N	N			N	N		N	N					N		N	
STATIC_TEXT			N	N			N	N		N	N					N		N	

**Dialog Control Property Descriptions**

Property	Type	Description
AUTOSCROLL	boolean	Control scrolls right automatically. Also scrolls down for Multiline Edit.
BORDER	boolean	Gives the control a border.
BOTTOM	integer	Sets the vertical position of the bottom edge of the control.
CONTROL_TEXT	string	Text displayed by the control.
DEFAULT	boolean	Button has heavy emphasis.
GROUPSTART	boolean	Used primarily to group radio buttons.
HEIGHT	integer	Sets the vertical size of the control.
HORZSCROLL	boolean	Control has a horizontal scroll bar.

HPSID	integer	Allows the template to recognize the control.
LEFT	integer	Sets the position of the control.
MARGIN	boolean	Gives the control a border.
MULTIPLE_SELECTION	boolean	Multiple items can be selected.
PASSWORD	boolean	Text is invisible.
READONLY	boolean	Text cannot be altered.
TABSTOP	boolean	User can tab to the control.
VERTSCROLL	boolean	Control has a vertical scroll bar.
WIDTH	integer	Sets the horizontal width of the control.
WORDWRAP	boolean	Text automatically wraps.

### Using Linked-to Variables

The control types in the following sections can have a linked-to clause. The linked-to clause is always optional. The linked-to clause must always specify a template variable.

#### CHECKBOX

The button is checked if the linked variable contains the value "TRUE," otherwise it is unchecked. After the dialog is closed, the variable contains either the value "TRUE" or "FALSE," depending on whether the user checked or unchecked the button.

#### RADIO\_BUTTON

The button is checked if the linked variable contains the HPSID of the RADIO\_BUTTON. After the dialog is closed, the variable contains the HPSID of the button only if it is checked. Normally, you link several mutually exclusive RADIO\_BUTTONS to a single variable. Then, the variable will contain the HPSID of the RADIO\_BUTTON that the user selects before closing the dialog.

#### EDIT\_FIELD

The text displayed in the EDIT\_FIELD is taken from the linked variable. After the dialog is closed, the linked variable contains the modified text.

#### MULTILINE\_EDIT

The text displayed in the MULTILINE\_EDIT is taken from the linked variable. After the dialog is closed, the linked variable contains the modified text.

#### LISTBOX

The linked variable must be list. The LISTBOX is populated using the items of the list variable. When the dialog is closed, the variable is modified to contain a list of the selected items from the LISTBOX.



If the LINKED\_TO variable has not been assigned a value, it defaults to the LOCAL variable.

### Using the TCDIALOG Variable to Set the Properties of the Dialog Window

TurboCycler creates a local variable named TCDIALOG upon entry into a dialog block. This variable represents the dialog window itself, and the following properties can be set using this variable:

- LEFT, for setting the left edge of the dialog
- BOTTOM, for setting the bottom edge of the dialog
- TOP, for setting the top edge of the dialog
- WIDTH, for setting the width of the dialog
- HEIGHT, for setting the height of the dialog
- CONTROL\_TEXT, for setting the caption displayed on the dialog title bar

### Processing the Dialog

After using MAKE statements to define the dialog, use the SHOW DIALOG statement to display the dialog to the user. When you close the dialog, TurboCycler creates a local variable named TCEVENTSOURCE that contains the HPSID of the push button that closed the dialog.





If you are running Microsoft Windows 2000, if you press the *Esc* key, the TCEVENTSOURCE variable is assigned the value of IDCANCEL.

## Template Block Features

### Template Block Features

The template language involves the following procedures:

- [Declaring Local Variables from Any Block](#)
- [Making Any Block a Callable Procedure](#)

### Declaring Local Variables from Any Block

Any block can declare local variables immediately after the first line of the block. Each local variable must be given an initial value. Here are two examples:

```
HIERARCHY "Test"
USES LOCAL \[X = 1, Y = "Hello"\]
// statements
ENDHIERARCHY

TEXT "Test" NAMED X
USES LOCAL \[Y = QUERY NAME OF X\]
// statements
ENDTEXT
```

Local variables are destroyed when the block exits. A local variable with the same name as a global variable hides the global variable. USE...ENDUSE blocks cannot declare local variables.

### Making Any Block a Callable Procedure

You can turn any block into a callable procedure simply by prefixing the block with the procedure keyword:

```
PROCEDURE
HIERARCHY "Test"
// statements
ENDHIERARCHY

PROCEDURE
RULE "Test" NAMED X
// statements
ENDRULE
```

Procedures are not part of any module, and they will only execute when called from elsewhere in the template. Use a CALL statement to call a procedure:

```
CALL PROCEDURE "Test"
// or...
SET PROC = "Test"
CALL PROCEDURE PROC
```

Arguments can also be passed. The procedure declares formal parameters as follows:

```
PROCEDURE REQUIRES \[X, Y, Z\]
HIERARCHY "Test"
// statements
ENDHIERARCHY
```

The parameters become local variables to the block and can be referenced immediately, even in the block header:

```
PROCEDURE REQUIRES \[ARULE, AVIEW\  
HIERARCHY "Test1"  
USES LOCAL \[X = QUERY NAME OF ARULE\  
// statements  
ENDHIERARCHY  
  
PROCEDURE REQUIRES \[RULENAME\  
RULE "Test2" NAMED RULENAME  
// statements  
ENDRULE
```

The CALL statement passes actual arguments:

```
CALL PROCEDURE "Test1" PASSING \[MYRULE, MYVIEW\  
]
```

Arguments are passed by reference only, so only currently defined variables can be passed as arguments. If the procedure alters the value of a parameter, the value of the corresponding argument is also altered.

Do not give two different procedures the same name?only the first will be recognized since there is no "module" concept with procedures.

Procedures can be called from USE and HIERARCHY blocks only. This restriction enforces the concept that other block types (rule and window, for example) cannot functionally decompose.

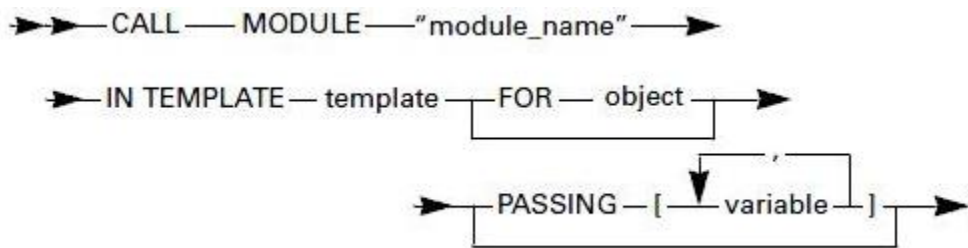
## Other Statements

You can also use these statements in building your application with the templates:

- [CALL Statement](#)
- [CALL SERVICE Statement](#)
- [SELECT Statement](#)
- [APPEND and REMOVE Statements](#)
- [TRAVERSE Statement](#)
- [UPDATE MESSAGE Statement](#)
- [SHOWMESSAGE Statement](#)
- [SET Statement](#)

### CALL Statement

The CALL statement allows one template to call another. Its syntax is:



This statement executes a module in a template for an object, just as if you passed the object to the template. Although the syntax allows the object to be optional, TurboCycler Standard Edition flags it as an error. This is a forward-looking language feature that supports the execution of a template without any repository object. The called template begins execution at the USE block that the FOR object names.

The "module\_name" and template parameters must be string expressions identifying the module name and template title (not its file name) to be run. The object parameter reduces to a repository object that the called template processes.

The optional PASSING clause describes a list of variables common to both templates. You can use each variable for both input and output.

A template can call itself. This is more efficient in space and time than calling another template, but keep the following points in mind:

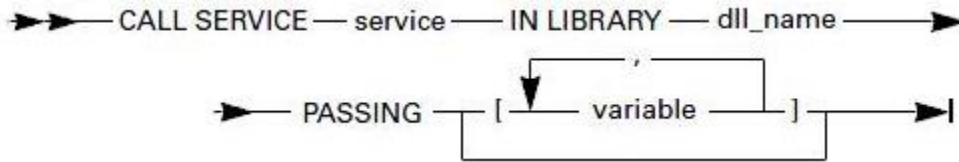
- The user can run any module in a template for any USE...ENDUSE blocks. Because the template cannot hide its modules from the user, you must document which modules can be generated from each object type the template accepts and instruct the user to configure the template from the setup dialog initially. The template must enforce the proper use of modules through programming.
- A template calling itself shares variables passed to it but shares no others. Each invocation of the template creates new instances of its variables.

An error results if the requested template does not exist, does not have the requested module, or does not accept the object type passed. In the case of a CALL statement, a USE ANY block is not considered a match for the object type.

The CALL MODULE statement is only allowed in USE...ENDUSE and HIERARCHY...ENDHIERARCHY statements.

### CALL SERVICE Statement

The CALL SERVICE statement supports user exits. A template can request a service from a dynamic link library (DLL). By using this call, the template and the DLLs can share variables.



You can initiate user exits to interact with the user or to implement additional functionality beyond the previous capabilities of the templates language.

The "C" header file (TCSDK.H) defines the application program interface for writing user exits. Sample user exit code can be found in C:\appbuilder\template\samples\userexit. You must put the user exit DLLs in the TurboCycler template directory on the development workstation.

### SELECT Statement

TurboCycler Release 2.2.0.3.1 includes a selection construct for evaluating arbitrary case expressions. The syntax is:

```
SELECT
CASE expr1 :
// statements
ENDCASE

CASE expr2 :
// statements
ENDCASE
.
.
.
DEFAULTCASE :
// statements
ENDCASE
ENDSELECT
```

### APPEND and REMOVE Statements

The APPEND statement appends an item to a list variable:

#### **APPEND expression TO list\_var**

If list\_var is not a list variable, it becomes one. The statement can be used to compose lists containing arbitrary expressions. The REMOVE statement removes an item from a list, collapsing subsequent items:

#### **REMOVE list\_var ( index )**

If the index is out of bounds, an error is generated.

### TRAVERSE Statement

The purpose of the TRAVERSE statement is to eliminate any explicit navigation logic from the template and to transform a recursive template algorithm into a finite automation. Use this statement to enable the template to navigate a hierarchy by specifying an explode path and a series of procedures that should be called at each "node."

```
TRAVERSE root_object
explode_path
ENDTRAVERSE
```

Here is an example showing the explode path:

```

USE RULE( ROOT )
TRAVERSE ROOT
\->VIEW VIA RELATION OWNS_VIEW : "doView"
->->FIELD VIA RELATION VIEW_INCLUDES : "doField"
->->RECURSE VIA RELATION VIEW_INCLUDES : "doView"
\->RECURSE VIA RELATION USES_RULE : "doRule"
ENDTRAVERSE
ENDUSE

```

In the example, the template is interested in finding all views, fields, and rules that are under the root rule. This is done by showing how the template should explode the hierarchy under the root object. In this case, the RECURSE statement is also used?it always refers to objects under the parent having the same type and related by way of the given relationship. It also specifies that the explosion should be repeated for these objects.

Therefore, the first RECURSE refers to the VIEW object because the parent object (having one less arrow) is a VIEW. Furthermore, it says that for each VIEW found, all the explode lines leading to (and including) the RECURSE should be repeated.

Note the following:

```

\->VIEW VIA RELATION OWNS_VIEW : "foo"
->->FIELD VIA RELATION VIEW_INCLUDES : "foo"
->->RECURSE VIA RELATION VIEW_INCLUDES : "foo"

```

Here all fields under the first view and all fields under every view under the first view are seen. Compare this to:

```

\->VIEW VIA RELATION OWNS_VIEW : "foo"
->->RECURSE VIA RELATION VIEW_INCLUDES : "foo"
->->FIELD VIA RELATION VIEW_INCLUDES : "foo"

```

where only fields under the first view are seen. However, views descending from the first view are seen here, just as in the first example. The last item on the line is the name of the procedure that should be called whenever this node is seen. The procedure is called with four arguments, so it must declare four parameters:

```

PROCEDURE "foo" REQUIRES \[PAR, KID, LINK, LEVEL\]
// any block type

```

where:

PAR	=	the parent object
KID	=	the child object
LINK	=	the relation between the parent and the child
LEVEL	=	the depth in the hierarchy, starting at 1



You cannot start one TRAVERSE while another is active. The purpose is to specify the complete explode path in the first one so there is no need to nest them. You can execute a STOP TRAVERSE statement from within a called procedure to prevent exploding children of the current child object to stop traversing a particular "branch" after the template has decided it is no longer interested in those children.

Here is an example of a template:

```

TEMPLATE "Test traverse"
DESCRIPTION "" ENDDescription
USAGES
USE FUNCTION (INFUNCTION)

SET FILENAME = "c:
\\
tra.out"
TRAVERSE INFUNCTION
\->PROCESS VIA RELATION REFINES_INT0 : "test"
->->RULE VIA RELATION IS_DEFINED_BY : "test"
->->->COMPONENT VIA RELATION USES_COMPONENT : "test"
->->->->RECURSE VIA RELATION COMPONENT_USES_COMPONENT : "test"
->->->RECURSE VIA RELATION USES_RULE : "test"
->->RECURSE VIA RELATION REFINES_INT0 : "test"
ENDTRAVERSE
ENDUSE
ENDUSAGES

PROCEDURE REQUIRES \[P, C, REL, DEPTH\]
FLATFILE "test" NAMED FILENAME
SET Z = TYPEOF(C) + "" + (QUERY NAME OF C)
UPDATE MESSAGE Z
FOR I = 1 TO DEPTH \{$( "\t" )\}
ENDFOR
\{$(TYPEOF(C)):$ (QUERY NAME OF C)$ ( "\n" )\}
ENDFLATFIL

```

### UPDATE MESSAGE Statement

Templates can optionally indicate progress by posting messages directly on the work-in-progress dialog using the UPDATE MESSAGE statement:

```
UPDATE MESSAGE "Still going..."
```

TurboCycler changes the message to the module name of each block as it enters that block.

### SHOWMESSAGE Statement

A function that is available for posting simple message boxes to the user. The syntax is:

```
SHOWMESSAGE (caption, message, buttons, icon)
```

The dialog has caption as its title and displays a message inside the dialog. The dialog can have one of the following values for buttons:

#### Dialog Buttons

Buttons	Value
OK	1
OK CANCEL	3
RETRY CANCEL	5
ABORT RETRY IGNORE	6
YES NO	7
YES NO CANCEL	8

Optionally, the dialog can use an icon:

#### Dialog Icons

Icon Style	Value
Hand	1
Question	2
Exclamation	3
Asterisk	4
Information	5
Query dialog	6
Warning dialog	7
Error dialog	8

The function returns the button pressed to close the dialog. It has one of the following values:

#### **Function Return Buttons**

Button	Value
CANCEL	0
OK	1
ENTER	2
ABORT	3
RETRY	4
IGNORE	5
YES	6
NO	7



A value of 8 is returned if an error occurs during the SHOWMESSAGE operation.

## **SET Statement**

The following statement for setting properties is available:

```
SET property OF object = value
```

This statement can be used in WINDOW blocks to set properties of window controls. This is an alternative to using the HAVING clause of the MAKE statement. These properties can also be retrieved using the QUERY expression. TurboCycler automatically creates a local variable named TCWINDOW upon entry to a WINDOW block. This variable refers to the window itself and can be used, for example, to set the CAPTION, ENTER\_KEY, and CLOSE\_KEY of the window:

```
WINDOW "Test" NAMED TEST
// window statements
SET CAPTION OF TCWINDOW = "Foobar"
ENDWINDOW
```

## **Supporting Statements and Expressions**


The last part of the template language contains language flows for the following:

- [Common Statement](#)

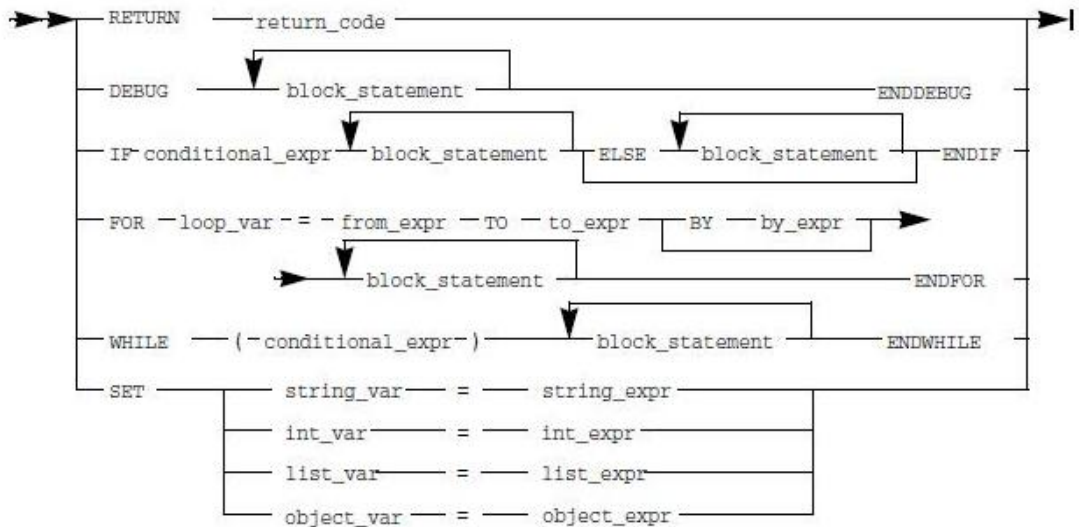
- [Conditional Expression \(conditional\\_expr\)](#)
- [Integer Expression \(int\\_expr\)](#)
- [List Expression \(list\\_expr\)](#)
- [Object Expression \(object\\_expr\)](#)
- [String Expression \(string\\_expr\)](#)

### Common Statement

The common statement is a sequence of template language statements that originates in several block statements, such as USAGES, WINDOW, RULE, and HIERARCHY.

 The flow diagrams indicate that the common statement refers back only to the originating block statement, not indiscriminately to any other block statement. You must return to the block where you started.

For example, if you originate in the RULE block, the scope of the "block statement" includes this common statement and that particular rule statement directing you here. The block statement function, including this common statement, supports recursive execution.



### RETURN

Keyword to end the execution of a template for the passed-in object. Through the execution of this statement, you can pass back a specifiable return code. See "return\_code" for an explanation of the return codes.


#### return\_code

Either an integer expression or string expression that indicates the reason for terminating the template. When the return code is:

- An integer, the number is not displayed. Zero means successful completion. Nonzero signifies an error, so any module that depends on this execution should not continue.
- A string expression, it is displayed and indicates an error.

### DEBUG...ENDDEBUG

A pair of keywords that surround one or more statements to be analyzed. DEBUG produces output including a trace of the executed instructions, branch conditions, variable assignments, and repository queries. You cannot change or select the DEBUG output. Debug records output in a file specified in the `DEBUG_FILE` option in the Tools> Workbench Options, TurboCycler set of options. You must set the fully qualified path and file name for your DEBUG file. You can also select to write the output only to a file. This will prevent the output from being displayed on the Analysis window in the Workbench.

 HPS.INI does not include the `DEBUG_FILE` parameter.

### block\_statement

A variable reference back to the block statement that invoked this common statement. The originator is one of the following block statements: usages block, rule block, window block, or hierarchy block.  
The block statement consists of this common statement and any unique instructions that are part of the block statement that invoked this common statement.  
A block statement and this common statement together can execute recursively, as the flow diagrams show.

### **IF...ENDIF**

Keywords that surround a sequence of block statements. The IF control conditionally executes the statements that follow. If the conditional expression is true, the block statement that immediately follows it is executed. The ELSE statement or the ENDIF indicate the end of the block. If the conditional statement is false, the ELSE statement is executed if it is present.

#### **conditional\_expr**

A conditional expression the program tests for validity. If true, the next block statements are run; if false, the ELSE block statements are run. (The template language does not have an explicit THEN.) Refer to [Conditional Expression \(conditional\\_expr\)](#) for the syntax.

### **ELSE**

An alternative logic path that the program follows when the conditional expression of the IF is not true.

### **FOR...ENDFOR**

Keywords that surround a sequence of block statements. The FOR loop control executes a sequence of instructions repetitively for a specified number of iterations. The range of the "from\_expr" (from expression) and the TO "to\_expr" (to expression) defines the number of loop iterations. You can increment the count of iterations by a value other than one with the BY "by\_expr" (by expression).

#### **loop\_var**

Loop variable (an integer variable) that is the counter for the FOR loop.

#### **from\_expr**

From expression (an integer expression) that sets the starting count for the FOR loop counter.

### **TO**

Keyword that specifies the end-point value that limits loop repetitions.

#### **to\_expr**

To expression (an integer expression) that sets the ending count for the FOR loop counter.

### **BY**

Keyword that introduces the value for incrementing the loop count. Otherwise, the increment defaults to counting by one.

#### **by\_expr**

By expression (an integer expression) that sets the incrementing value for the FOR loop counter.

### **WHILE?ENDWHILE**

Keywords that surround a sequence of block statements. The WHILE loop control executes a sequence of instructions repetitively until defined conditions are met. The "(conditional\_expr)" parameter defines the conditions.

#### **(conditional\_expr)**

A conditional expression, enclosed in required parentheses, on which this program evaluates and acts. While the conditional expression remains true, the following block statements are executed. Refer to [Conditional Expression \(conditional\\_expr\)](#) for the syntax.

### **SET**

A keyword that assigns values to your variables, including string, integer, list, and object variables.

=

The equal sign that is the assignment operator.



**string\_expr**

A string expression for string assignment. Refer to [String Expression \(string\\_expr\)](#).

**int\_expr**

An integer expression for integer assignment. Refer to [Integer Expression \(int\\_expr\)](#).

**list\_expr**

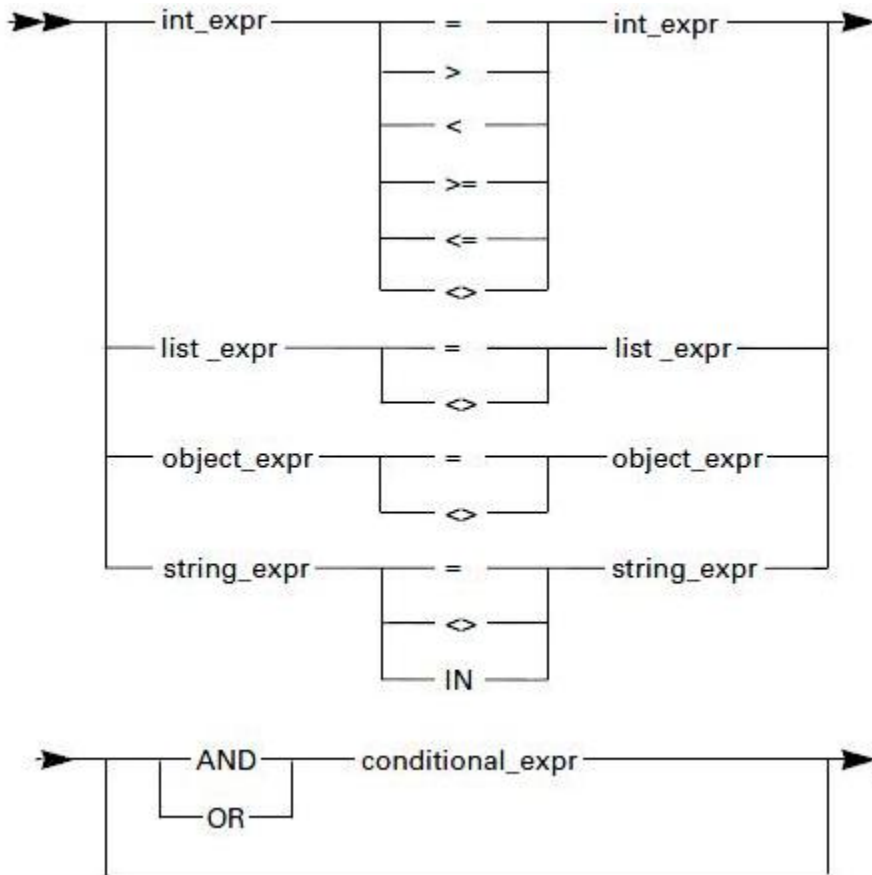
A list expression for a list assignment. Refer to [List Expression \(list\\_expr\)](#).

**object\_expr**

An object expression for an object assignment. Refer to [Object Expression \(object\\_expr\)](#).

**Conditional Expression (conditional\_expr)**

A conditional expression combines two expressions with a logical operator; the resultant combination evaluates to true or false. This evaluation can determine the sequence of programming execution. Expressions can be integers, lists, objects, and strings. Refer to [Logical and Arithmetic Operators](#) for comparison results.



**int\_expr**

Any integer expression, as defined in [Integer Expression \(int\\_expr\)](#).

**list\_expr**

Any list expression, as defined in [List Expression \(list\\_expr\)](#). Lists are equal when they have identical children in the same order.

**object\_expr**

Any object expression, as defined in [Object Expression \(object\\_expr\)](#). Objects are equal when they refer to identical repository objects.

### string\_expr

Any string expression, as defined in [String Expression \(string\\_expr\)](#). Strings are equal when their characters match exactly.

### AND/OR

Keywords that specify either an additional conditional expression or an alternative conditional expression that enhances the previous qualification.

### conditional\_expr

A conditional expression that can be added as another or an alternative conditional expression. (This usage supports recursive invocation of conditional expressions.)

### Logical and Arithmetic Operators

The following operators are used in conditional expressions:

=	equal
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
<>	not equal to
IN	contained in or embedded within another string

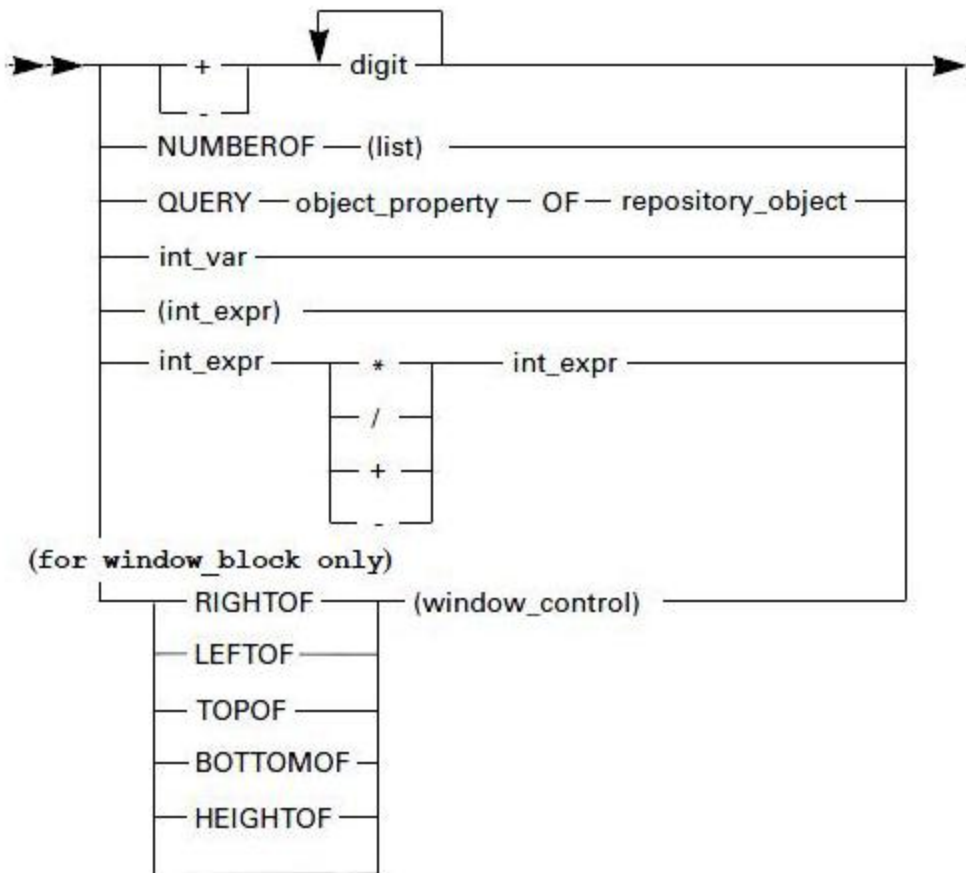
[Conditional Expressions](#) shows examples of conditional expressions and how they evaluate.

### Conditional Expressions

Expression	Evaluation
where X is 1 and Y is 2:	
x = y	False
y = x	False
x <> y	True
where x is Field X and y is View X:	
x = y	False
x <> y	True
where x = "TurboCycler" and y = "Turbo":	
x = y	False
x <> y	True
x IN y	False
y IN x	True

### Integer Expression (int\_expr)

An integer expression is a combination of whole number values, functions, or operators that evaluate to a whole number.



**+ and -**

Positive and negative arithmetic operators that act on the following "digit" parameter.

**digit**

Any numeral of one character.

**NUMBEROF**

A keyword that counts and reports the number of items in the list specified by the "(list)" parameter.

**(list)**

A list parameter (list expression) that identifies the list whose items are to be counted. Refer to [List Expression \(list\\_expr\)](#) for more about list expressions.

**QUERY**

A keyword that retrieves information for the repository object that the "repository\_object" parameter names.

**object\_property**

A parameter that describes the property to be retrieved for the specified object type. Some object properties return strings, others integers. Only object properties having and returning integer values are valid for the integer expression. Refer to [Object Types and Properties](#) for details.

**OF**

A keyword that identifies the repository\_object being queried.

**repository\_object**

An object expression that defines the repository object for which numeric information is being obtained.

**int\_var**

An integer variable in the template that has an integer value.

**(int\_expr)**

Any integer expression, as defined in [Integer Expression \(int\\_expr\)](#). The parentheses are optional, but you can use them to improve readability and to resequence the order of operators.

**int\_expr**

An integer expression that you can use with the following arithmetic operators to derive calculated values.

\* / + -

Arithmetic operators that function with the surrounding integer expressions and evaluate to an integer value.

**RIGHTOF/LEFTOF/TOPOF/BOTTOMOF/HEIGHTOF/WIDTHOF**

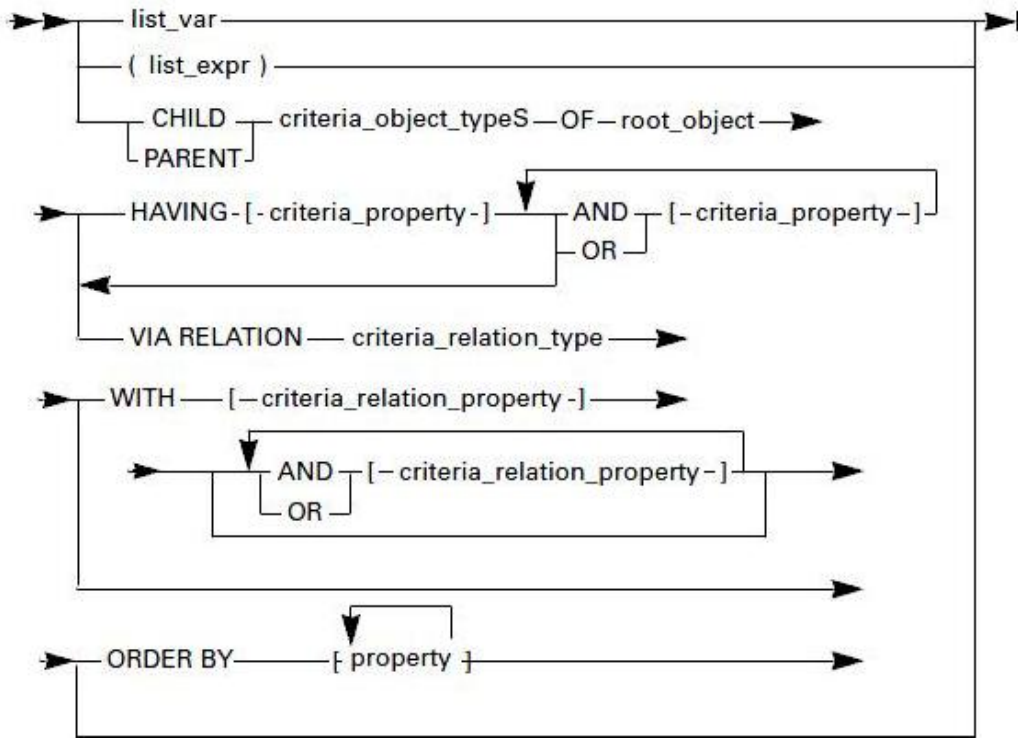
Limited to window block statements, these keywords retrieve the dimensions of window control parameters.

**(window\_control)**

A window control variable that refers to a window that the MAKE or CONTAIN statement creates. Refer to [WINDOW Block](#).

**List Expression (list\_expr)**

A list expression refers to a group of objects in the repository. Use this expression to obtain a list of objects whose properties match your selection criteria.



**list\_var**

A list variable in the template that refers to an object in the repository.

**(list\_expr)**

Any list expression enclosed in parentheses to improve readability.

## CHILDPARENT

Keywords that retrieve a list of children or parents of a given object. These keywords serve as the repository interface to promote navigation from object to object.

### **criteria\_object\_type S**

The object types of the children or parents to be retrieved from the list. The trailing character S signifies a list of objects is to be retrieved. The plural form of any object type supported by the Information Model is valid, such as RULES, WINDOWS, VIEWS, and FIELDS. (Note that plurals are constructed by putting the suffix S on the base word, with these exceptions: ENTITIES, PROCESSES, and LOGICAL\_PROCESSES.) Refer to [Object Types and Properties](#) for a list of object types.

## OF

A keyword that introduces the root\_object parameter as the point to begin navigation.

### **root\_object**

An object expression that refers to the object from which you want to start navigation.

## HAVING

A keyword that specifies properties the list of children or parent must have.

### **[...]**

A set of brackets enclosing the values of properties that must be met for the retrieved objects.

### **criteria\_property**

Criteria property (an object property expression) that specifies the values that the properties of the retrieved child and parent objects must have. Refer to [Object Types and Properties](#) for a complete list.

## AND/OR

Keywords that specify either an additional criteria\_property or an alternative criteria\_property that modifies the previous qualification.

## VIA RELATION

A keyword that specifies the relationship type between children or parents. This is a required parameter.

### **criteria\_relation\_type**

Criteria relation type (a relation type expression) that specifies the type of relationship to the children and parent objects. Refer to [Relationship Types and Properties](#) for a detailed list.

## WITH

A keyword that specifies required properties of the relationship (with children or parents) to the object.

### **criteria relation property**

Criteria relation type (a relation property expression) that specifies the values the properties of the relationship to the retrieved child/parent objects must have. Refer to [Relationship Types and Properties](#) for a detailed list.

## ORDER BY

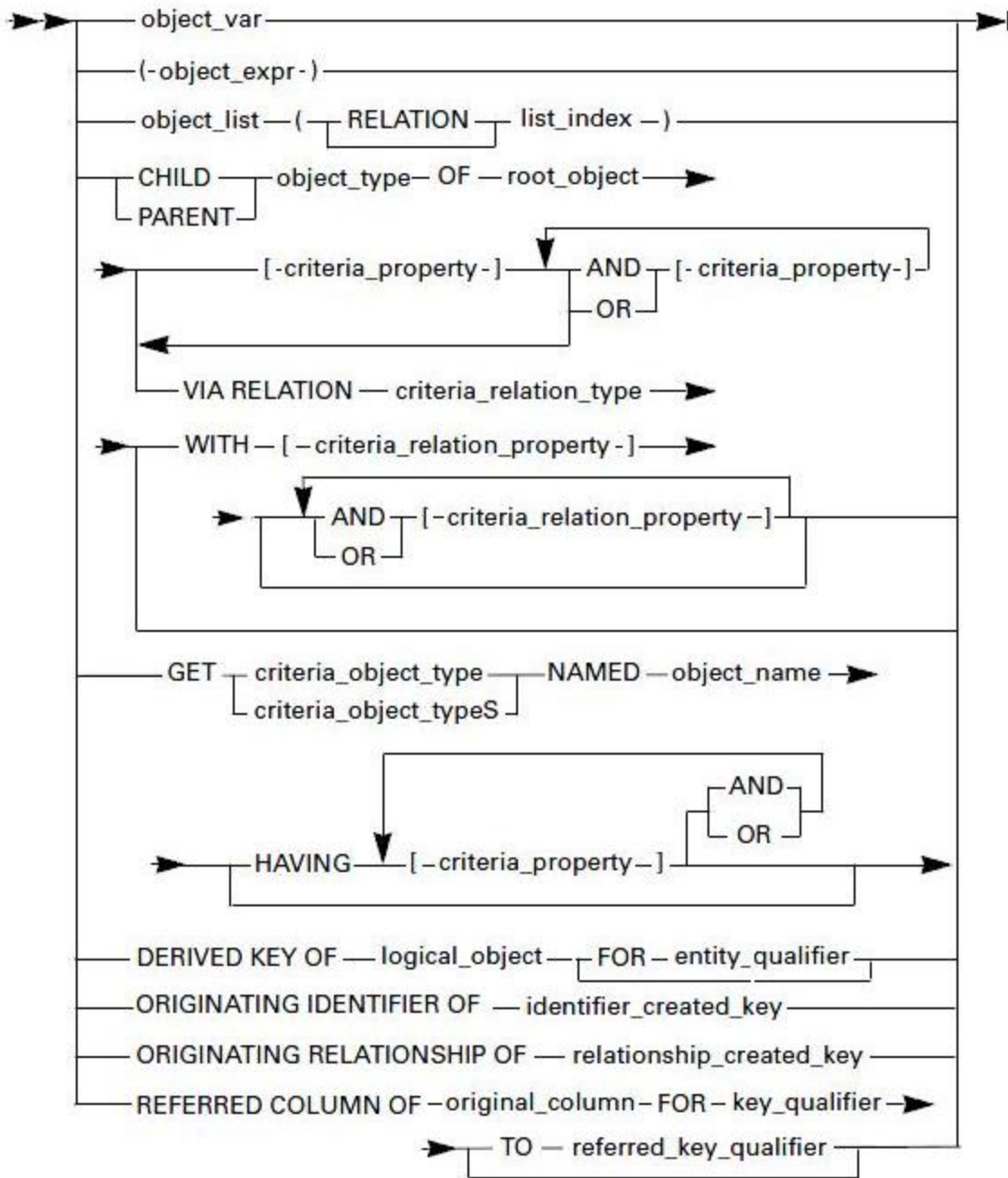
Sorts the list by the properties specified. Properties can apply to either the entity or the relationship.



TurboCycler automatically sorts lists by SEQUENCE\_NUMBER. Do not specify ORDER BY [SEQUENCE\_NUMBER].

## Object Expression (object\_expr)

An object expression refers to an object in the repository. Using this expression, you can obtain various properties about that object.



The last four flow lines address traceability data that the forward engineering and reverse engineering processes create.

**object\_var**

An object variable in the template that refers to an object in the repository.

**(object\_expr)**

Any object expression that can be enclosed in parentheses to improve readability.

**object\_list**

A list expression that represents a list of objects in the repository, which you index for a specific item in the list. The index is the offset of the item being accessed.

**[...]**

A set of brackets that index into a list of objects.

## **RELATION**

A keyword that refers to a relationship to the specified object in the repository, not to the object itself. Use this expression to obtain sequence number information among other data.

### **list\_index**

An integer expression that is the index into the object list. The first item in a list has an index value of one (1).

## **CHILD/PARENT**

Keywords that retrieve a list of children or parents of a given object. These keywords serve as the repository interface to promote navigation from object to object. They return only the first qualified object encountered. Refer to [Sample of a Hierarchy Diagram Coded as an Object Expression](#) for a coding example using CHILD and related parameters.

### **object\_type**

An object type of the child or parent to be retrieved. Refer to [Object Types and Properties](#) for a list of object types.

## **OF**

A keyword that introduces the root\_object parameter as the navigation starting point.

### **root\_object**

An object expression that refers to the object from which you want to start navigation.

### **[...]**

A set of brackets that enclose the values of properties the child/parent object must have.

### **criteria\_property**

Criteria property (an object property expression) that requires the child or parent object to have the specified value for the property. Refer to [Object Types and Properties](#) for a detailed list. Examples of the format for this statement are PROPERTY = string\_expr and PROPERTY = int\_expr.

## **AND/OR**

Keywords that specify either an additional criteria\_property or an alternative criteria\_property that modifies the previous qualification.

## **VIA RELATION**

A keyword that specifies the relationship type between the child or parent. This is a required parameter. Refer to [Sample of a Hierarchy Diagram Coded as an Object Expression](#) for a coding example using VIA RELATION and related parameters.

### **criteria\_relation\_type**

Criteria relation type specifies the type of relationship to the child/parent object. Refer to [Relationship Types and Properties](#) for details.

## **WITH**

A keyword that specifies requisite properties of the relationship (with a child or parent) to the object. Refer to [Sample of a Hierarchy Diagram Coded as an Object Expression](#) for a coding example using WITH and related parameters.

### **criteria\_relation\_property**

Criteria relation type (a relation property expression) that specifies the values that the properties of the relationship to the child/parent object must have.

## **GET**

A keyword that specifies repository objects to be retrieved. You can make conditional queries of repository objects.

### **criteria\_object\_type**

A property that specifies the type of the object being retrieved. Refer to [Object Types and Properties](#) for a list of valid object types. When the criteria\_object\_type parameter is not plural, this expression retrieves a single repository object. When criteria\_object\_type is plural, the expression

retrieves any number of objects of the specified type from the repository.

#### **NAMED**

A keyword that identifies the "object\_name" parameter.

#### **HAVING**

A keyword that specifies properties the child or parent must have. Use the HAVING clause to select a subset of these objects. Refer to [Sample of a Hierarchy Diagram Coded as an Object Expression](#) for a coding example using HAVING and related parameters.

#### **object\_name**

A string expression that specifies the name of the object to be retrieved. The object\_name parameter can optionally contain a single asterisk \* as the last character. When accessing the members of the resulting list, the RELATION subscript modifier has no meaning.

#### **DERIVED KEY OF**

A keyword that obtains the key that implements an identifier or a relationship in the repository.

To understand key usage with relationships or identifiers, you must understand how forward engineering handles keys. Forward engineering begins with an entity relationship diagram, which consists of entities, attributes, relationships, and identifiers. The forward engineering process turns those objects into a database diagram consisting of tables, keys, and columns. As part of this activity, forward engineering links keys to relationships to produce a foreign key and links keys to identifiers to produce a primary or index key. Consequently, the following TurboCycler functions occur:

#### ***DERIVED KEY OF Relationship (provides a foreign key)***

DERIVED KEY OF Identifier FOR entity\_qualifier  
(provides a primary or index key)

#### **logical\_object**

An object expression that is either an ERD relationship or an identifier. If a relationship, it is not qualified by the entity\_qualifier and returns a foreign key. If an identifier, it must be qualified by its owning entity and returns a primary or an index key.

#### **FOR**

A keyword that introduces the "entity\_qualifier" parameter. This phrase applies only to an identifier object that is querying for a primary or index key.

#### **entity\_qualifier**

An object expression that names the entity for which an identifier is being sought.

#### **ORIGINATING IDENTIFIER OF**

A keyword that starts the query for an identifier by searching through its primary or index key. This function is the reverse of the "DERIVED KEY OF logical\_object FOR entity\_qualifier" function. In this case, you supply the key and receive the identifier object.

#### **identifier\_created\_key**

An object expression that names the primary or index key from which you receive the corresponding identifier object.

#### **ORIGINATING RELATIONSHIP OF**

A keyword that initiates the query for a relationship by searching through its foreign key. This function is the reverse of the [logical\\_object](#). In this case, you supply the key and receive the relationship object.

#### **relationship\_created\_key**

An object expression that names the foreign key through which you obtain the corresponding relationship object.

#### **REFERRED COLUMN OF**

A keyword that returns a column object that is referred by another column when two tables are connected by a foreign key.

To understand referred column queries between tables with keys, you must understand how tables, keys, and columns are structured in database diagrams. If two tables have columns connected by a foreign key, you can extract column objects about one by using the key data from the other. For instance, consider the following example:



### ***REFERRED COLUMN OF columnf FOR forkey***

In this example, you supply a foreign key column (columnf) and the foreign key (forkey) and receive the referred column object of the foreign key by which columnf is referred. Now, here is another example:

### ***REFERRED COLUMN OF columnp FOR prikey TO forkey***

In this example, you supply the primary key column (columnp) and the primary or index key (prikey) referred by the foreign key (forkey) and receive the foreign key column object that is referring to columnp. Refer to [Example of REFERRED COLUMN OF coding](#) for a visual representation of these examples of the REFERRED COLUMN OF keyword.

#### **original\_column**

An object expression that names the column object from which the search begins.

#### **FOR**

A keyword that names the key\_qualifier through which searching starts.

#### **key\_qualifier**

An object expression that identifies the key that is searched for the desired response object.

#### **TO**

A keyword that names the foreign key, as described in the second example of the REFERRED COLUMN OF keyword.

#### **referred\_key\_qualifier**

An object expression that names the foreign key, as described in the second example of the REFERRED COLUMN OF keyword.

#### **Additional information**

[Sample of a Hierarchy Diagram Coded as an Object Expression](#) shows a coding example of the CHILD entity type, HAVING, VIA RELATION, and WITH.

#### **Deleting Objects**

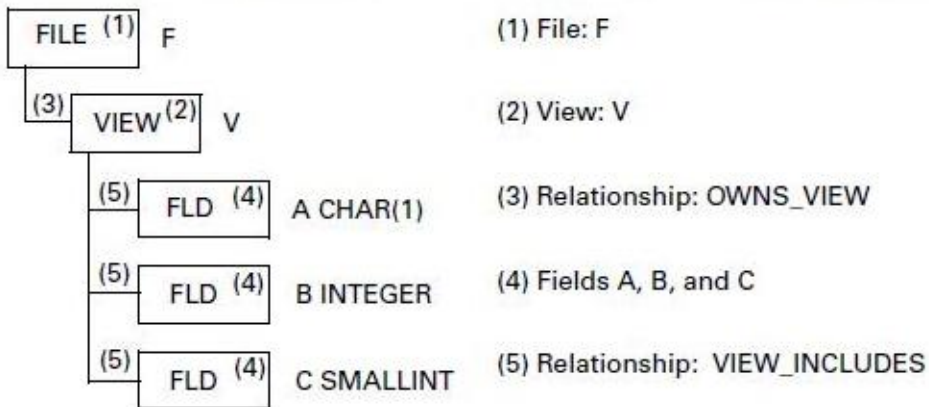
A keyword that specifies repository objects to be deleted.

#### **Delete (object\_expr)**

#### ***Sample of a Hierarchy Diagram Coded as an Object Expression***

### Sample of a Hierarchy Diagrammer

### Essential elements of the hierarchy



### Sample of the equivalent coded object expression for view V:

CHILD VIEW OF F  
VIA RELATION OWNS\_VIEW  
WITH [SEQUENCE\_NUMBER = 10] AND [TYPE = "DATA"]

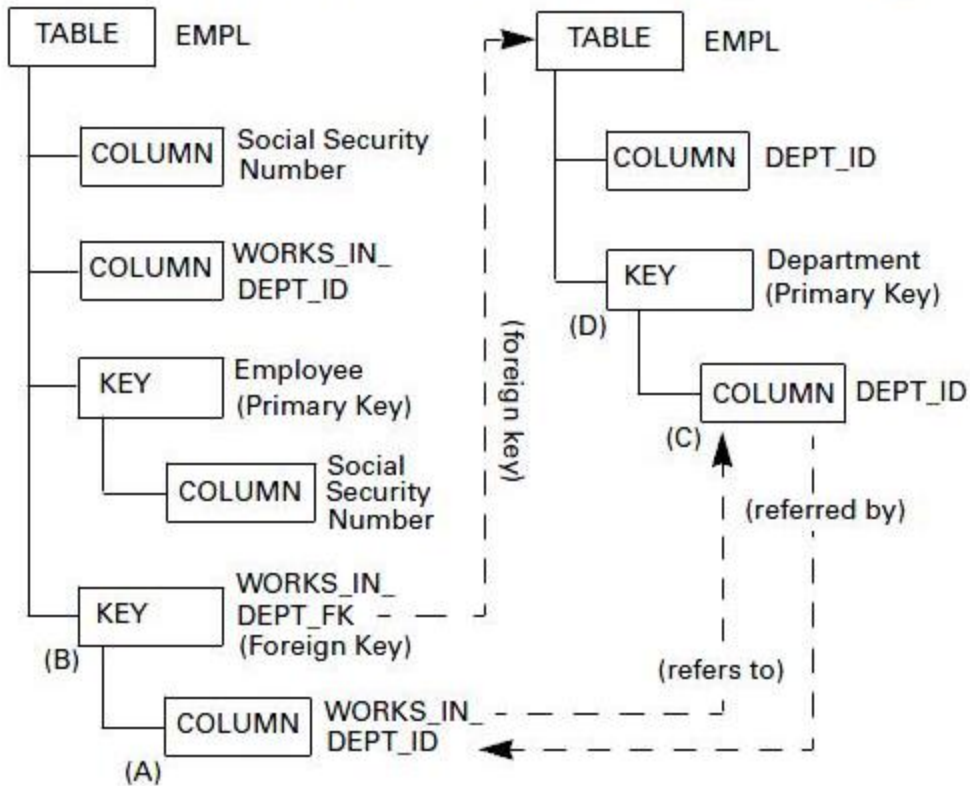
### Sample of the equivalent coded object expression for field B:

CHILD VIEW OF V  
HAVING [TYPE = "INTEGER"]  
VIA RELATION VIEW\_INCLUDES

[Example of REFERRED COLUMN OF coding](#) shows an example of a diagram and flow for the REFERRED COLUMN OF statement.

*Example of REFERRED COLUMN OF coding*

**Example of tables and columns connected by foreign key relationship:**



**Sample of template language coding:**

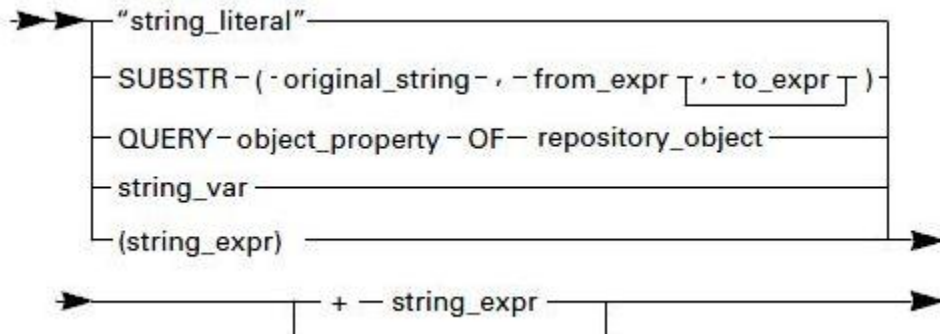
- (1) REFERRED COLUMN OF *columnf* (A) FOR *forkey* (B)
- (2) REFERRED COLUMN OF *columnp* (C) FOR *prikey* (D) TO *forkey* (B).

**Results:**

- (1) Statement 1 queries A and B, returning column C.
- (2) Statement 2 queries C and D, returning column A.

**String Expression (string\_expr)**

A string expression refers to a string of characters.



**"string\_literal"**

A character string enclosed in double quotes.

**Lexical analysis of string literals**

The backslash character "\" is the escape sequence in string literals. Supported escape sequences are described in [Escape Sequences](#).

### Escape Sequences

Characters	Description
\	Backslash
\t	Horizontal tab (HT)
\v	Vertical tab (VT)
\b	Backspace (BS)
\n	Newline (LF)
\r	Carriage return (CR)
\f	Form feed (FF)
\a	Alert (BEL)
\"	Double quote
\xhhh	Hex number (all three digits required)

If any other character follows a backslash, the backslash is ignored.



Use the double backslash for file names. For example: SET myDirectory = "G:  
HPS  
"

### SUBSTR (...)

A keyword that retrieves a specified character substring from the "original\_string" parameter.

#### original\_string

A string expression that identifies the character string from which the SUBSTR retrieves a substring.

#### from\_expr

An integer expression that indicates the starting point of the "original\_string" parameter of retrieved data.

#### to\_expr

An integer expression that indicates the end point of the "original\_string" parameter of retrieved data.

### QUERY

A keyword that retrieves property information for the repository object that is named by the repository\_object parameter.

#### object\_property

A parameter that describes the property to be retrieved for the specified object type. Some object properties return strings, others integers. Only the object properties having and returning string values are valid for the object expression. [Object Types and Properties](#)

### OF

A keyword that identifies the "repository\_object" parameter being queried.

#### repository\_object

An object expression that defines the repository object for which character string information is being obtained.

#### string\_var

A string variable in the template that has a character value.

(string\_expr)

Any string expression, as defined in [String Expression \(string\\_expr\)](#). The parentheses are optional, but you can use them to improve readability.

+ string\_expr

A string expression that concatenates another string expression to the one just completed.

## Processing Symbols on an Open Drawing

A USE type is supported that allows a template to process the symbols on an open drawing. An example of a template that can be generated from an open ERD is:

```
USAGES
USE ERD Drawing ( _drw_ )
// statements
ENDUSE
ENDUSAGES
```

Here the variable *drw* refers to the open drawing. To obtain a list of the entities on the drawing, use:

```
SET ENTS = CONNECTED ENTITIES OF DRW
```

The variable ENTS contains the drawing symbols?not repository objects. Drawing symbols have the following properties:

- TYPE: The symbol type (ENTITY, for example).
- ASSOCIATED\_OBJECT: The associated repository object, if any.
- SELECTED: "TRUE" if the symbol is selected by the user; otherwise, it is "FALSE."
- FOCUSED: "TRUE" if the symbol was in focus; otherwise, it is "FALSE." (Only one symbol can have the focus.)

Here is an example of the use of the ENTS variable:

```
FOR I = 1 TO NUMBEROF( ENTS )
SET ENTITY = QUERY ASSOCIATED_OBJECT OF ENTS(I)
ENDFOR
```

A symbol can be connected to other symbols. Both boxes and lines are considered symbols without distinction. For the ERD example:

```
SET ENTITYSYMBOL = ENTS(1)

// Query for relationship lines coming from the entity
SET LINES = CONNECTED RELATIONSHIP_LINES OF ENTITYSYMBOL
VIA CONNECTION RELATION_OUT

// Query for relationship lines coming into the entity
SET LINES = CONNECTED RELATIONSHIP_LINES OF ENTITYSYMBOL
VIA CONNECTION RELATION_IN

SET REPO_OBJECT = QUERY ASSOCIATED_OBJECT OF LINES(1)
SET REPO_LINK = QUERY ASSOCIATED_OBJECT OF LINES(CONNECTION 1)
```

In the last two lines, repository objects of types RELATIONSHIP and IS\_RELATED\_VIA are returned, respectively. These statements have a very strong analogy to the TurboCycler CHILD...VIA RELATION and RELATION array subscript modifier statements?they are meant to work in the same way for drawing symbols.

[Entity Relationship Diagrammer](#) through [Window Flow Diagrammer](#) list the symbols associated with different types of diagrams.

### Entity Relationship Diagrammer

Symbol Name	Plural Name	Symbol Type
RELATION_OUT	n/a	connection
RELATION_IN	n/a	connection

SUBTYPE_OUT	n/a	connection
SUBTYPE_IN	n/a	connection
ENTITY	ENTITIES	box
BUSINESS_OBJECT	BUSINESS_OBJECTS	box
RELATIONSHIP_LINE	RELATIONSHIP_LINES	line
SUBTYPE_LINE	SUBTYPE_LINES	line
NOTE	NOTES	box

**State Transition Diagrammer**

Symbol Name	Plural Name	Symbol Type
IS_PRECONDITION	n/a	connection
IS_POSTCONDITION	n/a	connection
EVENT_CLAUSES	n/a	connection
STATE	STATES	box
EVENT	EVENTS	box
TRANSITION_LINE	TRANSITION_LINES	line

**Database Diagrammer**

Symbol Name	Plural Name	Symbol Type
REFERRED_OUT	n/a	connection
REFERRED_IN	n/a	connection
HAS_OUT	n/a	connection
HAS_IN	n/a	connection
TABLE	TABLES	box
KEY	KEYS	box
FOREIGN_KEY	FOREIGN_KEYS	box
INDEX_KEY	INDEX_KEYS	box
HAS_KEY_LINE	HAS_KEY_LINES	line
REFERRED_KEY_LINE	REFERRED_KEY_LINES	line

**Process Dependency Diagrammer**

Symbol Name	Plural Name	Symbol Type
DEPEND_OUT	n/a	connection
DEPEND_IN	n/a	connection
TRIGGER_OUT	n/a	connection
TRIGGER_IN	n/a	connection
EVENT_TRIGGER_IN	n/a	connection

EVENT_TRIGGER_OUT	n/a	connection
LOGICAL_PROCESS	LOGICAL_PROCESSES	box
DECISION	DECISIONS	box
TRIGGER_LINE	TRIGGER_LINES	line
DEPENDENCY_LINE	DEPENDENCY_LINES	line
EVENT_TRIGGER_LINE	EVENT_TRIGGER_LINES	line

#### **Window Flow Diagrammer**

<b>Symbol Name</b>	<b>Plural Name</b>	<b>Symbol Type</b>
FLOW_OUT	n/a	connection
FLOW_IN	n/a	connection
NEST_OUT	n/a	connection
NEST_IN	n/a	connection
EVENT_OUT	n/a	connection
EVENT_IN	n/a	connection
DETACH_OUT	n/a	connection
DETACH_IN	n/a	connection
RULE_INITIATES	n/a	connection
DECISION_OUT	n/a	connection
DECISION_IN	n/a	connection
DECISION_NEST_OUT	n/a	connection
PROCESS	PROCESSES	box
ENTRY_POINT	ENTRY_POINTS	box
DIALOG_UNIT	DIALOG_UNITS	box
WINDOW	WINDOWS	box
RULE	RULES	box
DECISION	DECISIONS	box
TERMINAL	TERMINALS	box
FLOW_LINE	FLOW_LINES	line
NESTED_FLOW_LINE	NESTED_FLOW_LINES	line
DETACHED_FLOW_LINE	DETACHED_FLOW_LINES	line
MESSAGE_FLOW_LINE	MESSAGE_FLOW_LINES	line

## **Functions for TurboCycler Developer's Kit**

The TurboCycler Developer's Kit comes with a number of commonly needed functions that you can use to help you build your templates.

- [Get and File Functions](#)
- [String Manipulation Functions](#)
- [Superseded Functions](#)

### **Get and File Functions**

The following functions are available:

- [GetConfiguration \(app, key\)](#)
- [GetEnvironment \(environment\\_variable\)](#)
- [GetHPSConfiguration \(app, key\)](#)
- [GetTime \(format\\_string\)](#)
- [isDefined \(variable\\_name\)](#)
- [NewFilename \( \)](#)
- [removeFile \(file\\_name\)](#)
- [ReadFlatFile\(pathname\)](#)

#### ***GetConfiguration (app, key)***

This function retrieves the value of the key parameter from the app parameter section of the file to which the environment variable TURBOINI points. You can use this function to place editable parameters in this file, rather than hard coding them in the template. Documentation provided with the template must explain what is required with respect to this file. Templates must share this file. If the entry cannot be found, an empty string is returned.

#### ***GetEnvironment (environment\_variable)***

This function retrieves the value of any environment\_variable. If the requested variable is not defined, an empty string is returned.

#### ***GetHPSConfiguration (app, key)***

This function retrieves the value of the key parameter from the registry. If the entry cannot be found, an empty string is returned.



The following sample key and values illustrate how one can set up the logic:

```
_HKEY_CURRENT_USER\Software\BluePhoenix\AppBuilder\IDWB\Tools\TurboCycler\Settings_ \\  
_ "BITMAP_DIR"="D:\AppBuilder\NT\RT\BMP" _ \\  
_ "MENU_FILE"="D:\AppBuilder\NT\RT\MENU" _
```

Then add a statement similar to the following:

```
SET gBITMAP_DIR = GetHPSConfiguration( "_" Tools \  
TurboCycler \  
Settings "_" , "_" BITMAP_DIR "_" ) \  
SET gMENU_FILE = GetHPSConfiguration( "_" Tools \  
TurboCycler \  
Settings "_" , "_" MENU_FILE "_" )
```

In Windows, the HPSINI file cannot be read using the GetHPSConfiguration function. However, AppBuilder can use the Registry for the TurboCycler GetHPSConfiguration. The following sample key and values illustrate how one can set up the logic:

```
HKEY_CURRENT_USERS\Software\BluePhoenix\AppBuilder\IDWB\Tool\TurboCycler\Settings  
"BITMAP_DIR"="D:\AppBuilder\NT\RT\BMP"  
"MENU_FILE"="D:\AppBuilder\NT\RT\MENU"  
Then, add a statement similar to the one below:  
SET gBITMAP_DIR = GetHPSConfiguration("Tools  
TurboCycler  
Settings", "BITMAP_DIR" )  
SET gMENU_FILE = GetHPSConfiguration("Tools  
TurboCycler  
Settings", "MENU_FILE" )
```

#### ***GetTime (format\_string)***

This function uses the format\_string parameter to generate a string containing the current date or time information. The format\_string is composed of text and, optionally, one or more of the conversion specifiers described in [Conversion Specifiers](#):

#### ***Conversion Specifiers***



Specifier	Description	Example
%a	Abbreviated weekday name	Sun
%A	Full weekday name	Sunday
%b	Abbreviated month name	Dec
%B	Full month name	December
%c	Date and time	Dec 02 1995 06:55:15
%d	Day of the month	02
%H	Hour of the 24-hour day	23
%I	Hour of the 12-hour day	11
%j	Day of the year, from 001	335
%m	Month of the year, from 01	12
%M	Minutes after the hour, from 00	55
%p	AM/PM indicator (AM)	AM
%S	Seconds after the minute, from 00	48
%U	Sunday week of the year, from 00	51
%w	Day of the week, from 0 for Sunday	0
%W	Monday week of the year from 00	43
%x	Date	Dec 02 1995
%X	Time	06:55:15
%y	Year of the century, from 00	95
%Y	Year	1995
%z	Time zone name, if any	EST
%%	Percent character	%

Using any other % character in the format string is illegal, and an empty string is returned.

#### ***isDefined (variable\_name)***

This function returns 1 if the variable\_name is defined as a variable in the template; otherwise, it returns 0. Use this function to determine whether the template is being executed by a user or another template.

#### ***NewFilename ( )***

This function generates a unique file name.

#### ***removeFile (file\_name)***

This function attempts to delete the file specified by the file\_name parameter. It returns 1 if successful or 0 if unsuccessful.

#### ***ReadFlatFile(pathname)***

TurboCycler reads in text files using the READFLATFILE (pathname) function. The function returns a string containing the text file. Only text files smaller than 64K can be read using this function.

### **String Manipulation Functions**



The first character in a string is at position one, not position zero.

#### ***strCenter (str, length, pad)***

Returns a string of length with str centered. Remaining spaces are filled using pad.

***strCopies (str, n)***

Returns a string of n concatenated copies of str.

***strCutLeft (str, length)***

Returns str with the leftmost length characters removed.

***strCutMiddle (str, start, length)***

Returns str with length characters starting at position start removed.

***strCutRight (str, length)***

Returns str with the rightmost length characters removed.

***strFilespec (str, "option")***

Returns the part of the file name in str specified by the option. The option can be: Drive, Dir, Name, or Extension. The "option" parameter must be of type string, otherwise an error will be received when trying to use the turbo cyler.

***strFind (needle, haystack, int n)***

Returns the position of the nth occurrence of needle in haystack.

***strInsert (needle, haystack, n)***

Inserts needle in haystack at position n.

***strLeft (str, length)***

Returns a string comprised of the leftmost length characters of str.

***strLength (str)***

Returns the number of characters in str.

***strLine (str, i)***

Returns the i the line in str.

***strLines (str)***

Returns the number of lines in str, separated by carriage returns.

***strLower (str)***

Converts str to lower case.

***strMiddle (str, start, length)***

Returns a string comprised of length character taken from str beginning at position start.

***strReplace (search, replace, str)***

Replaces each occurrence of the string search in str with the string replace. The search is case insensitive.

***strReverse (str)***

Returns a string comprised of the characters from str in reverse order.

***strRight (str, length)***

Returns a string comprised of the rightmost length character of str.

### ***strToken (str, delimiter\_set, n)***

Returns the nth token in str, using the characters in delimiter\_set as delimiters.

### ***strTokens (str, delimiter\_set)***

Returns the number of tokens in str separated by any of the characters in delimiter\_set.

### ***strTrim (str, "option")***

Returns str without leading or trailing white space. The option can be: Leading, Trailing, or Both. The "option" parameter must be of type string, otherwise an error will be received when trying to use the turbo cyclor.

### ***strUpper (str)***

Converts str to upper case.

### ***strWord (str, i)***

Returns the i the word in str.

### ***strWords (str)***

Returns the number of words in str, separated by white space.

## **Superseded Functions**

[Superseded Functions](#) lists functions and statements from previous releases of TurboCyclor that have since been superseded and are no longer used.

### ***Superseded Functions***

<b>For these functions:</b>	<b>Use:</b>
<ul style="list-style-type: none"><li>• RIGHTOF()</li><li>• LEFTOF()</li><li>• WIDTHOF()</li><li>• HEIGHTOF()</li></ul>	The new QUERY property statement instead.
<b>For these statements:</b>	<b>Use:</b>
POSITION WINDOW SIZE WINDOW	The SET property statement instead: SET BOTTOM OF TCWINDOW = 50 SET LEFT OF TCWINDOW = 50 SET WIDTH OF TCWINDOW = 200 SET HEIGHT OF TCWINDOW = 150

## **Template Samples**

The TurboCyclor Developer's Kit includes the following two sample templates to show you how to generate object hierarchies, rules, and a window:

- [SQL Delete Rule Sample](#) - part of the CRUD Rules template
- [Detail Display Rule and Window Sample](#) - part of the GUI Rules/Windows template

These samples demonstrate many of the features available in the template language. The samples are located in the directory AD\TEMPLATE\SAMPLES.



Source code for all the Standard Edition templates is included with the TurboCyclor Developer's Kit in the directory APPBUILDER\AD\TEMPLATES.

### **SQL Delete Rule Sample**

This sample is a part of the Standard Edition CRUD Rules template that generates an SQL Delete Rule for an AppBuilder file object. Generating a

completely functional SQL Delete Rule requires defining both the rule hierarchy and rule source code. Therefore, this sample template includes both a hierarchy module and rule module. Both modules have the same name, forcing the generation of both the hierarchy and rule. In the usages block, this template retrieves all the information it needs from the repository and sets up any local variables that multiple modules require. In this example, the process error checks the generated rule source code and rule hierarchy. You can generate an SQL Delete Rule without error-checking, but any errors would remain undetected until you prepare the generated rule. In the hierarchy block, the template generates the SQL Delete Rule hierarchy. Input and output view are both declared as children of the rule and as root-level objects. These declarations ensure that TurboCycler deletes all children not created from this template. In some cases, you might not want to do this. In the rule block, the template generates the source code for the SQL Delete Rule. It includes a comment area at the top saying that it was generated and specifies the name of the file for which it was generated. The placement of the { } (curly brace characters) is critical to generating properly spaced and indented rules source. A good example of this placement is the WHERE clause of the SQL ASIS.

```

TEMPLATE "SQL Delete Rule Sample"
DESCRIPTION
"Generates an SQL Delete Rule for an AppBuilder File
object."
ENDDescription

USAGES
USE FILE(MyFile)
REM --- Retrieve primary view information from repository \---;
SET MyPrimeView = CHILD VIEW OF MyFile VIA RELATION
OWNS_VIEW WITH \[VIEW_USAGE = "PRIMARY"\]
SET MyPrimeFields = CHILD FIELDS OF MyPrimeView
VIA RELATION VIEW_INCLUDES

REM --- Retrieve file information from repository \---;
SET MyFileName = QUERY NAME OF MyFile
SET MyFileImplementation = QUERY IMPLEMENTATION_NAME
OF MyFile
ENDUSE

USE ALL
REM --- Verify that we have a primary view \---;
IF MyPrimeView = 0
RETURN "The file does not have a primary view."
ENDIF

REM --- Verify that we have primary fields \---;
IF (NUMBEROF(MyPrimeFields)) = 0
RETURN "The primary view for the file does not have
any fields."
ENDIF

REM --- Verify that we have a file implementation name \---;
IF MyFileImplementation = ""
RETURN "The file does not have an implementation
name."
ENDIF

REM --- Verify that all primary fields have an
implementation name \---;
FOR I = 1 TO NUMBEROF(MyPrimeFields)
IF (QUERY IMPLEMENTATION_NAME OF MyPrimeFields(I))
= ""
SET MESSAGE = "The primary field " + (QUERY NAME
OF MyPrimeFields(I)) + " does not have an
implementation name."
RETURN MESSAGE
ENDIF
ENDFOR
ENDUSE
ENDUSAGES

REM \-----\
--- SQL Delete Rule Modules ---
\-----;

REM \-----;

```

```

HIERARCHY "SQL Delete Rule"

REM --- Create Strings for SQL Delete Rule Modules \---;
SET DeleteRuleName = (SUBSTR (MyFileName, 1, 19) ) +
"SQL_DEL"
SET DeleteRuleIV = DeleteRuleName + "_IV"
SET DeleteRuleOV = DeleteRuleName + "_OV"

REM --- Build Hierarchy for the SQL Delete Rule \---;
\> RULE DeleteRuleName \[DBMS = "DB2"\]
->-> FILE MyFileName VIA RELATION ACCESSES
\[SEQUENCE_NUMBER = 10\]
->-> VIEW DeleteRuleIV VIA RELATION OWNS_VIEW
\[VIEW_USAGE = "Input", SEQUENCE_NUMBER = 20\]
->-> VIEW DeleteRuleOV VIA RELATION OWNS_VIEW
\[VIEW_USAGE = "Output", SEQUENCE_NUMBER = 30\]

REM --- Build Hierarchy for the SQL Delete Rule Input View \---;
\> VIEW DeleteRuleIV
->-> VIEW MyPrimeView VIA RELATION VIEW_INCLUDES
\[SEQUENCE_NUMBER = 10\]

REM --- Build Hierarchy for SQL Delete Rule Output View \---;
\> VIEW DeleteRuleOV
->-> FIELD "RETURN_CODE" VIA RELATION VIEW_INCLUDES
\[SEQUENCE_NUMBER = 10\]

ENDHIERARCHY

RULE "SQL Delete Rule" NAMED DeleteRuleName
\{
*&-----<*&
*&\- Rule: $(DeleteRuleName) \-<*&
*&\- SQL Delete Rule for the File: $(MyFile). \-<*&
*&\- Automatically Generated by TurboCycler. \-<*&
*&-----<*&

SQL ASIS
DELETE FROM $(MyFileImplementation)
WHERE
\}

FOR I = 1 TO NUMBEROF(MyPrimeFields)
\{ $(QUERY IMPLEMENTATION_NAME OF MyPrimeFields(I)) =
:$(DeleteRuleIV).$(MyPrimeFields(I)) \}
IF I <> NUMBEROF(MyPrimeFields)
\{AND
\}
ENDIF
ENDFOR

\{
ENDSQL

IF SQLCODE = 0
SQL ASIS
COMMIT
ENDSQL
ENDIF

```

```

MAP SQLCODE TO RETURN_CODE
\}
ENDRULE

```

## Detail Display Rule and Window Sample

This sample is part of the Standard Edition GUI Rules/Windows template that generates a simple Detail Display rule and window with no application logic for an AppBuilder file object. Generating an executable detail display hierarchy requires defining a function/process hierarchy, rule hierarchy, window hierarchy, rule source code, and window panel. Therefore, the sample template includes two hierarchy blocks, a rule block, and a window block. All modules have the same name, which forces the generation of all of these blocks.

In the usages block, this template retrieves all information that it needs from the repository and sets up any local variables that multiple modules require. In this example, the process error checks the generated rule source code and window panel. If error-checking is not performed, you can generate a detail display window without any fields in it (if no fields were in the data view).

The use of the FOR loop over the fields in the data view clones the data view. This example also shows how to create any kind of hierarchy, including a function/process hierarchy. The sample rule block is relatively simple because it does not include any application logic.

In the window block, the template makes several controls, including a static text and edit field for every field in the data view. In doing this, a variable (MyBottom) stores the current value of the bottom position of the next window control. This template also creates a simple menu hierarchy and includes a bit map in the top left corner of the screen. It is important to understand how the template maintains the size and position of the window.

```

TEMPLATE "Detail Display Sample"
DESCRIPTION
"Generates a Detail Display Rule and Window for a
AppBuilder File Object."
ENDDescription

USAGES
USE FILE(MyFile)
REM --- Query repository for information needed by
blocks \---;
SET MyDataView = CHILD VIEW OF MyFile VIA RELATION
OWNS_VIEW WITH \[VIEW_USAGE = "DATA"\]
SET MyFields = CHILD FIELDS OF MyDataView VIA RELATION
VIEW_INCLUDES
SET MyFileName = QUERY NAME OF MyFile
ENDUSE

USE ALL
REM --- Verify there is a data view \---;
IF MyDataView = 0
RETURN "The file does not have a data view."
ENDIF

REM --- Verify the data view has fields \---;
IF (NUMBEROF(MyFields)) = 0
RETURN "The data view does not have any fields."
ENDIF

REM --- Set up Strings needed by blocks \---;
SET MyClonedDataView = (SUBSTR (MyFileName, 1, 19) ) +
"_CDV"
SET MyDetailRule = (SUBSTR (MyFileName, 1, 16) ) +
"_SAMPLE_DETAIL"
SET MyDetailWindow = (SUBSTR (MyFileName, 1, 16) ) +
"_SAMPLE_DETAIL"
SET MyProcessName = (SUBSTR (MyFileName, 1, 19) ) +
"_MAINT"
ENDUSE

ENDUSAGES

REM --- Detail Display Function/Process/Rule Hierarchies \---;

HIERARCHY "Detail Display"

REM --- Create the Function/Process hierarchy \---;

```

```

\-> FUNCTION MyFileName \[MENU_DESCRIPTION = MyFileName\]
->-> PROCESS MyProcessName \[MENU_DESCRIPTION = MyFileName
+ " Maintenance"\] VIA RELATION REFINES_INTO
\[SEQUENCE_NUMBER = 10\]

REM --- Attach a root rule to the Process \---;
\-> PROCESS MyProcessName
->-> RULE MyDetailRule VIA RELATION IS_DEFINED_BY
\[SEQUENCE_NUMBER = 10\]

REM --- Create the Detail Display Rule Hierarchy \---;
\-> RULE MyDetailRule
->-> WINDOW MyDetailRule VIA RELATION CONVERSES_WINDOW
\[SEQUENCE_NUMBER = 110\]
->-> VIEW "HPS_EVENT_VIEW" VIA RELATION OWNS_VIEW
\[SEQUENCE_NUMBER = 120, VIEW_USAGE = "WORK"\]

ENDHIERARCHY

REM --- Detail Display Rule Source Code \---;
RULE "Detail Display" NAMED MyDetailRule
\{
*<-----<
*>\- Rule: $(MyDetailRule) \-<*>
*>\- Detail Display Rule for the File: $(MyFile). \-<*>
*>\- Automatically Generated by TurboCycler. \-<*>
*>-----<

do while EVENT_SOURCE <> 'Exit'
map TIME to STATUS_TIME
map DATE to STATUS_DATE
converse window $(MyDetailRule)

caseof EVENT_SOURCE
case 'New'
clear $(MyDetailRule)
endcase

enddo
\}
ENDRULE

REM --- Detail Display Window Hierarchy \---;
HIERARCHY "Detail Display"

REM --- Create the Window Hierarchy \---;
\-> WINDOW MyDetailWindow
->-> VIEW MyDetailWindow VIA RELATION OWNS_VIEW
\[SEQUENCE_NUMBER = 10\]

REM --- Create the Window View Hierarchy \---;
\-> VIEW MyDetailWindow
->-> FIELD "STATUS_DATE" \[TYPE = "DATE", LENGTH = 4\]
VIA RELATION VIEW_INCLUDES \[SEQUENCE_NUMBER = 10\]
->-> FIELD "STATUS_TIME" \[TYPE = "TIME", LENGTH = 4\]
VIA RELATION VIEW_INCLUDES \[SEQUENCE_NUMBER = 20\]
->-> VIEW MyClonedDataView VIA RELATION VIEW_INCLUDES
\[SEQUENCE_NUMBER = 30\]

REM --- Clone the data view in the window view \---;
\-> VIEW MyClonedDataView
FOR X = 1 TO NUMBEROF(MyFields)
->-> FIELD (QUERY NAME OF MyFields(X)) VIA RELATION
VIEW_INCLUDES \[SEQUENCE_NUMBER = (10*X)\]
ENDFOR

ENDHIERARCHY

REM --- Detail Display Window Panel \---;
WINDOW "Detail Display" NAMED MyDetailWindow

```

```

SET MyCDV = GET VIEW NAMED MyClonedDataView

REM --- Make the Detail Display Menu \---;
\-> MENUITEM "&File"
->-> MENUITEM "&New" \[HPSID = "New"\]
->-> SEPARATOR
->-> MENUITEM "&Print" \[HPSID = "HPS_MENU_PRINT"\]
->-> SEPARATOR
->-> MENUITEM "E&xit" \[HPSID = "Exit"\]

\-> MENUITEM "&Edit"
->-> MENUITEM "&Cut" \[HPSID = "HPS_MENU_CUT"\]
->-> MENUITEM "Co&py" \[HPSID = "HPS_MENU_COPY"\]
->-> MENUITEM "&Paste" \[HPSID = "HPS_MENU_PASTE"\]

REM --- Set up area information \---;
SET MyWidth = 440
SET MyBottom = (20 + (29 * (NUMBEROF(MyFields) - 1)))

REM --- Place all edit fields \---;
FOR X = 1 TO NUMBEROF(MyFields)

MAKE STATIC_TEXT
HAVING \[HPSID = "ID" + (QUERY NAME OF MyFields(X)),
LEFT = 20,
TEXT = (QUERY SCREEN_LITERAL OF MyFields(X)),
BOTTOM = MyBottom,
WIDTH = 150,
HEIGHT = 19,
FONT = "SWISS8"\]
ENDMAKE

REM --- PROCEDURE TO DETERMINE WIDTH OF FIELD \---;
SET FieldType = QUERY TYPE OF MyFields(X)
SET FieldLength = QUERY LENGTH OF MyFields(X)
IF FieldType = "INTEGER"
IF FieldLength = 15
SET ControlWidth = 60
ELSE
SET ControlWidth = 100
ENDIF
ELSE
IF FieldType = "DATE" OR FieldType = "TIME"
SET ControlWidth = 100
ELSE
SET ControlWidth = 7 * FieldLength
IF ControlWidth < 30
SET ControlWidth = 30
ENDIF
ENDIF
ENDIF

MAKE EDIT_FIELD edit
HAVING \[HPSID = (QUERY NAME OF MyFields(X)),
LEFT = 175,
BOTTOM = MyBottom,
HEIGHT = 19,
WIDTH = ControlWidth,
FONT = "SWISS8",
TABSTOP = "TRUE"\]
LINKED TO MyFields(X) OF MyCDV
ENDMAKE

SET MyBottom = MyBottom - 29

IF (RIGHTOF(edit) + 20) > MyWidth
SET MyWidth = RIGHTOF(edit) + 20
ENDIF

```



```
ENDFOR

REM --- Reset the bottom to be above all edit fields \---;
SET MyBottom = 20 + (29 * NUMBEROF(MyFields)) + 15

REM --- Place Bitmap/Status Information Above Edit Fields \---;

MAKE BITMAP
HAVING \[HPSID = "HPS", LEFT = 20,
BOTTOM = MyBottom, WIDTH = 50,
HEIGHT = 50, FILE = "DEFAULT.BMP"\]
ENDMAKE
MAKE PROTECTED_EDIT_FIELD
HAVING \[HPSID = "STATUS_DATE", LEFT = 80,
BOTTOM = MyBottom + 31, HEIGHT = 19,
WIDTH = 60, FONT = "SWISS8",
TABSTOP = "FALSE"\]
LINKED TO (GET FIELD NAMED "STATUS_DATE")
ENDMAKE

MAKE PROTECTED_EDIT_FIELD
HAVING \[HPSID = "STATUS_TIME", LEFT = 80,
BOTTOM = MyBottom, HEIGHT = 19,
WIDTH = 60, FONT = "SWISS8",
TABSTOP = "FALSE"\]
LINKED TO (GET FIELD NAMED "STATUS_TIME")
ENDMAKE

SET MyBottom = MyBottom - 2
MAKE RECTANGLE
HAVING \[HPSID = "Line", LEFT = 0,
BOTTOM = MyBottom, WIDTH = MyWidth,
HEIGHT = 1\]
ENDMAKE

SET MyBottom = MyBottom + 50

REM --- Size and Position the window \---;
SIZE WINDOW MyWidth BY (MyBottom + 45)
```

```
POSITION WINDOW AT 50, 50  
  
ENDWINDOW
```

## TurboCycler Repository Types and Properties

Repository Types and Properties describes the object types in the Information Model that TurboCycler can access or generate. The Information Model defines the objects that an AppBuilder repository can contain. Repository objects include entities and relationships. The Information Model describes the entity types and relationship types they can have with each other. These objects have properties that you can query or set. Some properties have a set of values called a domain.

Repository Types and Properties describe the types of objects, the relationships between them, and the properties of each that you can use in template language statements.

This includes:

- [Object Types and Properties](#)
- [Entity Types and Properties](#)
- [Entities and Relationships](#)
- [Relationship Types and Properties](#)

This chapter begins with a short description of how to access repository information. The next section shows a short example with template language statements using repository types and properties. Then, object and relationship types and their corresponding properties are described. Repository Types and Properties ends with a list of the domains for properties that have them.

## Repository Object Type Query Sample

A number of TurboCycler template language statements can access the repository and perform queries. For example, you can query the type of a repository object using the TYPEOF (object\_expression) function:

```
SET X = GET FIELD NAMED "RETURN_CODE"  
SET Y = TYPEOF( X ) // Y = "FIELD"
```



All object query statements return lists that are automatically sorted by sequence number.

## Template Language Statement Sample

The following template language statement navigates to the data view of FILEA from within a generation template:

```
SET DataView = CHILD VIEW OF FILEA  
VIA RELATION OWNS_VIEW  
WITH [VIEW_USAGE = "DATA"]
```

The next example defines the child object type (VIEW) as an object type. VIEW is a valid entity type with three properties: an implementation name, a name, and a system ID.

```
VIEW  
IMPLEMENTATION_NAME  
NAME  
SYSID
```

The next example defines the child relationship type OWNS\_VIEW as a relationship type having two properties: sequence number and view usage. The VIEW\_USAGE property has a domain, ViewUsage.

```
OWNS_VIEW  
SEQUENCE_NUMBER  
VIEW_USAGE: ViewUsage
```

The ViewUsage domain controls the values in the following example:

```
ViewUsage
ALTERNATE CANDIDATE
DATA FOREIGN
INPUT INPUT/OUTPUT
OUTPUT PRIMARY
WORK
```

You can find a view with VIEW\_USAGE 'output' or VIEW\_USAGE 'input/output'. Use the following syntax to find a view:

```
[VIEW_USAGE = "INPUT/OUTPUT"]
```

or

```
[VIEW_USAGE = "INPUT"]
```

## Object Types and Properties

Object types have properties that can be set or queried. The properties of the object type are described after the object. [TurboCycler Entity Types](#) lists the supported entity types.

### *TurboCycler Entity Types*

<a href="#">APPLICATION_CONFIGURATION</a>	<a href="#">KEY</a>
<a href="#">ASSUMPTION</a>	<a href="#">LANGUAGE</a>
<a href="#">ATTRIBUTE</a>	<a href="#">LOCATION</a>
<a href="#">BITMAP</a>	<a href="#">LOGICAL_PROCESS</a>
<a href="#">BITMAP_IMPLEMENTATION</a>	<a href="#">MACHINE</a>
<a href="#">BUSINESS_OBJECT</a>	<a href="#">MIGRATION</a>
<a href="#">CELL</a>	<a href="#">OPPORTUNITY</a>
<a href="#">COLLECTION</a>	<a href="#">ORGANIZATION</a>
<a href="#">COLUMN</a>	<a href="#">PANEL</a>
<a href="#">COMPONENT</a>	<a href="#">PARTITION</a>
<a href="#">COMPONENT_FOLDER</a>	<a href="#">PHYSICAL_EVENT</a>
<a href="#">CONTEXT_DIAGRAM</a>	<a href="#">PROCESS</a>
<a href="#">DATABASE</a>	<a href="#">RELATIONSHIP</a>
<a href="#">DATA_FLOW</a>	<a href="#">REPORT</a>
<a href="#">DATA_STORE</a>	<a href="#">REQUIREMENT</a>
<a href="#">DATA_TYPE</a>	<a href="#">RULE</a>
<a href="#">DEVELOPMENT_PROJECT</a>	<a href="#">SECTION</a>
<a href="#">DRAWING</a>	<a href="#">SERVER</a>
<a href="#">ENTITY</a>	<a href="#">SET</a>
<a href="#">ER_VIEW</a>	<a href="#">SOFTWARE_PACKAGE</a>
<a href="#">EVENT</a>	<a href="#">STATE</a>
<a href="#">EXTERNAL_AGENT</a>	<a href="#">SUCCESS_FACTOR</a>

<a href="#">FIELD</a>	<a href="#">SYMBOL</a>
<a href="#">FILE</a>	<a href="#">SYSTEM</a>
<a href="#">FORM</a>	<a href="#">TABLE</a>
<a href="#">FSDM_SCHEMA</a>	<a href="#">TRANSITION</a>
<a href="#">FSDM_VALUE</a>	<a href="#">VALUE</a>
<a href="#">FUNCTION</a>	<a href="#">VERSION</a>
<a href="#">GOAL</a>	<a href="#">VIEW</a>
<a href="#">HELP</a>	<a href="#">WINDOW</a>
<a href="#">IDENTIFIER</a>	<a href="#">WINDOW_CONTENT</a>
<a href="#">INFORMATION_NEED</a>	

All object types have the properties OBJECT\_TEXT and OBJECT\_KEYWORDS. For example:

**SET X = QUERY OBJECT\_TEXT OF Y**

The result of this expression is to store the TEXT of object Y as a string in variable X.

Additionally, there are audit properties that are common to all objects in the Workgroup Repository. [Workgroup Repository Audit Properties](#) is a list of the audit properties and their character restrictions:

**Workgroup Repository Audit Properties**

Properties	Character Restriction
FWY_PROJECT	char 10
FWY_USER	char 10
FWY_DATE	char 8
FWY_TIME	char 8
FWY_LOCKOWNER	char 10
FWY_OWNER	char 10

**Audit Properties**

All entities have the read-only audit properties listed in [Universal Read-Only Audit Properties](#).

**Universal Read-Only Audit Properties**

CHANGE_NUMBER	OBJECT_KEYWORDS
OBJECT_TEXT	OWNER
PROJECT	REMOTE_DATE_CREATED
REMOTE_TIME_CREATED	REMOTE_CREATED_BY
REMOTE_DATE_MAINTAINED	REMOTE_TIME_MAINTAINED
REMOTE_MAINTAINED_BY	

## Entity Types and Properties

The following table provides information on entities and their properties. Each named entity is also hyperlinked to [Relationships](#), which lists the relationships for each parent-child relationships for each entity.

**Entities and their Properties**

---

Name	Properties	Domain Values
<a href="#">APPLICATION_CONFIGURATION</a>	Name Sysid	
<a href="#">ASSUMPTION</a>	ASSUMPTION_DESCRIPTION TYPE PRIORITY ACTUAL_START_DATE PLANNED_START_DATE CONFIDENCE_LEVEL RISK	TYPE Properties: <ul style="list-style-type: none"> <li>• FINANCIAL</li> <li>• BUSINESS</li> <li>• TECHNICAL</li> </ul> PRIORITY Values: <ul style="list-style-type: none"> <li>• MEDIUM</li> <li>• LOW</li> <li>• HIGH</li> </ul> CONFIDENCE_LEVEL Values: <ul style="list-style-type: none"> <li>• LOW</li> <li>• MEDIUM</li> <li>• HIGH</li> </ul> RISK Values: <ul style="list-style-type: none"> <li>• MEDIUM</li> <li>• LOW</li> <li>• HIGH</li> </ul>
<a href="#">ATTRIBUTE</a>	DERIVATION NAME SYSID	DERIVATION Values: <ul style="list-style-type: none"> <li>• DERIVED</li> <li>• FUNDAMENTAL</li> </ul>
<a href="#">BITMAP</a>	IMPLEMENTATION_NAME NAME SYSID TYPE	TYPE Domain values: <ul style="list-style-type: none"> <li>• BITMAP</li> <li>• ICON</li> <li>• JPEG</li> </ul>
<a href="#">BITMAP_IMPLEMENTATION</a>	NAME SYSID TYPE X_RESOLUTION Y_RESOLUTION	TYPE Domain value: <ul style="list-style-type: none"> <li>• BITMAP</li> <li>• ICON</li> <li>• JPEG</li> </ul>
<a href="#">BUSINESS_OBJECT</a>	BUSINESS_OBJECT_DESCRIPTION NAME SYSID TYPE	TYPE values: <ul style="list-style-type: none"> <li>• REFERENTIAL</li> <li>• TRANSACTIONAL</li> </ul>
<a href="#">COLLECTION</a>	NAME NATURE NUMBER_OF_ELEMENTS NUMBER_OF_GROUPS SYSID TYPE	NATURE values: <ul style="list-style-type: none"> <li>• LOGICAL</li> <li>• PHYSICAL</li> <li>• RELATIONAL</li> </ul>

<a href="#">COLUMN</a>	<p>           AVE_LENGTH            IMPLEMENTATION_NAME            LENGTH            NAME            SCALE            SYSID            TYPE         </p>	<p>TYPE Domain values:</p> <ul style="list-style-type: none"> <li>• BOOLEAN</li> <li>• CHARACTER</li> <li>• DATE</li> <li>• DECIMAL</li> <li>• GRAPHIC_CHARACTER</li> <li>• IMAGE</li> <li>• INTEGER</li> <li>• MIXED_CHARACTER</li> <li>• OBJECT_REFERENCE</li> <li>• PICTURE</li> <li>• TEXT</li> <li>• TIME</li> <li>• TIMESTAMP</li> <li>• VARCHAR</li> </ul>
<a href="#">COMPONENT</a>	<p>           DBMS_USAGE            ENVIRONMENT            EXECUTION_MODE            IMPLEMENTATION_NAME            LANGUAGE            NAME            SOURCE_FILE            SYSID         </p>	<p>DBMS_USAGE values:</p> <ul style="list-style-type: none"> <li>• DB2</li> <li>• NOT_APPLICABLE</li> <li>• DLI</li> </ul> <p>ENVIRONMENT values:</p> <ul style="list-style-type: none"> <li>• CICS</li> <li>• CICS_BATCH</li> <li>• IMS</li> <li>• MVS_BATCH</li> <li>• PC</li> <li>• PC_SYSTEM</li> <li>• PC_USER</li> </ul> <p>EXECUTION_MODE Domain values:</p> <ul style="list-style-type: none"> <li>• HAS_SUBROUTINE</li> <li>• NO_SUBROUTINE</li> </ul> <p>LANGUAGE values:</p> <ul style="list-style-type: none"> <li>• ASSEMBLER</li> <li>• COBOL</li> <li>• C</li> <li>• JAVA</li> <li>• PLI</li> </ul>
COMPONENT_FOLDER	<p>           COMPONENT_TYPE            COMPONENT_FOLDER_DESCRIPTION         </p>	
<a href="#">CONTEXT_DIAGRAM</a>	<p>SOURCE_FILE</p>	
<a href="#">DATABASE</a>	<p>           IMPLEMENTATION_NAME            MACHINE_NAME            NAME            SYSID            TYPE         </p>	<p>TYPE Domain values:</p> <ul style="list-style-type: none"> <li>• DB2</li> <li>• DB2/UDB</li> <li>• ORACLE</li> <li>• MS-SQLSERVER</li> </ul>
<a href="#">DATA_FLOW</a>	<p>           TYPE            IMPLEMENT_AS         </p>	<p>TYPE Domain values:</p> <ul style="list-style-type: none"> <li>• CONTROL</li> <li>• BOTH</li> <li>• DATA</li> </ul> <p>IMPLEMENT_AS values:</p> <ul style="list-style-type: none"> <li>• UNKNOWN</li> <li>• SCREEN</li> <li>• REPORT</li> </ul>

<a href="#">DATA_STORE</a>	DATA_STORE_DESCRIPTION NAME SYSID	
<a href="#">DATA_TYPE</a>	FRACTION LENGTH NAME SYSID TYPE	TYPE Domain values: <ul style="list-style-type: none"> <li>• BOOLEAN</li> <li>• CHARACTER</li> <li>• DATE</li> <li>• DECIMAL</li> <li>• GRAPHIC_CHARACTER</li> <li>• IMAGE</li> <li>• INTEGER</li> <li>• MIXED_CHARACTER</li> <li>• OBJECT_REFERENCE</li> <li>• PICTURE</li> <li>• TEXT</li> <li>• TIME</li> <li>• TIMESTAMP</li> <li>• VARCHAR</li> </ul>
<a href="#">DEVELOPMENT_PROJECT</a>	ACTUAL_END_DATE ACTUAL_START_DATE ESTIMATED_COST ESTIMATED_ROI PLANNED_START_DATE PLANNED_END_DATE RISK STATUS PRIORITY TECHNICAL_COMPLEXITY TYPE	RISK Domain values: <ul style="list-style-type: none"> <li>• MEDIUM</li> <li>• LOW</li> <li>• HIGH</li> </ul> STATUS Domain values: <ul style="list-style-type: none"> <li>• PLANNED</li> <li>• ACTIVE</li> </ul> PRIORITY Domain values: <ul style="list-style-type: none"> <li>• MEDIUM</li> <li>• LOW</li> <li>• HIGH</li> </ul> TECHNICAL_COMPLEXITY Domain values: <ul style="list-style-type: none"> <li>• MEDIUM</li> <li>• LOW</li> <li>• HIGH</li> </ul>
<a href="#">DRAWING</a>	NAME SYSID TYPE	TYPE Domain values: <ul style="list-style-type: none"> <li>• DATABASE_DESIGN</li> <li>• DATA_STORE_VS_ENTITY</li> <li>• ENTITY_RELATIONSHIP</li> <li>• ENTITY_VS_LOCATION_1</li> <li>• ENTITY_VS_LOCATION_2</li> <li>• FUNCTION_VS_ENTITY</li> <li>• FUNCTION_VS_LOCATION</li> <li>• MATRIX</li> <li>• ORGANIZATION_VS_ENTITY</li> <li>• ORGANIZATION_VS_PROCESS</li> <li>• PROCESS_DEPENDENCY</li> <li>• PROCESS_VS_ENTITY</li> <li>• STATE_TRANSITION</li> <li>• SYSTEM_VS_DATA_STORE</li> <li>• SYSTEM_VS_PROCESS</li> <li>• WINDOW_FLOW</li> </ul>

<a href="#">ENTITY</a>	ACTIVITY_PERIOD AVERAGE_DELETES AVERAGE_INSERTS AVERAGE_UPDATES EXPECTED_ROWS MAXIMUM_ROWS MINIMUM_ROWS NAME SYSID TYPE	ACTIVITY_PERIOD Domain values: <ul style="list-style-type: none"> <li>• HOUR</li> <li>• MINUTE</li> <li>• SECOND</li> <li>• DAY</li> <li>• QUARTER</li> <li>• YEAR</li> <li>• MONTH</li> <li>• WEEK</li> </ul> TYPE Domain values: <ul style="list-style-type: none"> <li>• ASSOCIATIVE</li> <li>• CHARACTERISTIC</li> <li>• INTERSECTION</li> <li>• KERNEL</li> </ul>
<a href="#">ER_VIEW</a>	ER_VIEW_DESCRIPTION SOURCE_FILE SOURCE_FILE	
<a href="#">EVENT</a>	CLASS EVENT_DESCRIPTION NAME SYSID TYPE	CLASS Domain values: <ul style="list-style-type: none"> <li>• TEMPORAL</li> <li>• EXTERNAL</li> <li>• INTERNAL</li> </ul>
<a href="#">EXTERNAL_AGENT</a>	EXTERNAL_AGENT_DESCRIPTION TYPE	TYPE Properties: <ul style="list-style-type: none"> <li>• SYSTEM</li> <li>• ROLE</li> <li>• ORGANIZATION</li> </ul>
<a href="#">FIELD</a>	FRACTION IMPLEMENTATION_NAME LENGTH MAXIMUM MINIMUM NAME PICTURE_CLAUSE REF SCREEN_LITERAL SCREEN_PICTURE SHORT_SCREEN_LITERAL SYSID TYPE	TYPE Domain values: <ul style="list-style-type: none"> <li>• CHARACTER</li> <li>• DATE</li> <li>• DECIMAL</li> <li>• GRAPHIC_CHARACTER</li> <li>• IMAGE</li> <li>• INTEGER</li> <li>• MIXED_CHARACTER</li> <li>• PICTURE</li> <li>• TEXT</li> <li>• TIME</li> <li>• TIMESTAMP</li> <li>• VARCHAR</li> </ul>
<a href="#">FILE</a>	IMPLEMENTATION_NAME NAME SYSID TYPE	TYPE Domain values: <ul style="list-style-type: none"> <li>• DB2</li> <li>• DB2/UDB</li> <li>• ORACLE</li> <li>• MS-SQLSERVER</li> </ul>



<a href="#">FORM</a>	ENVIRONMENT BASE COUNTRY_LANGUAGE SOURCE_FILE	ENVIRONMENT Domain values: <ul style="list-style-type: none"> <li>• WINDOWS</li> <li>• AFP</li> </ul> BASE Domain values: <ul style="list-style-type: none"> <li>• FALSE</li> <li>• TRUE</li> </ul> COUNTRY_LANGUAGE Domain values: <ul style="list-style-type: none"> <li>• US_ENGLISH</li> <li>• UK_ENGLISH</li> </ul>
<a href="#">FSDM_SCHEMA</a>	FSDM_SCHEME_DESCRIPTION MUTUALLY_EXCLUSIVE NAME SYSID	MUTUALLY_EXCLUSIVE Domain values: <ul style="list-style-type: none"> <li>• FALSE</li> <li>• TRUE</li> </ul>
<a href="#">FSDM_VALUE</a>	FSDM_VALUE_DESCRIPTION NAME SYSID	
<a href="#">FUNCTION</a>	CHILD_MENU MENU_DESCRIPTION NAME SYSID WORKSTATION_GROUP	CHILD_MENU Domain values: <ul style="list-style-type: none"> <li>• MENU_BAR</li> <li>• NO_MENU</li> <li>• PULL_DOWN</li> </ul>
<a href="#">GOAL</a>	GOAL_DESCRIPTION TYPE PRIORITY RISK ACTUAL_START_DATE PLANNED_START_DATE CONFIDENCE_LEVEL TARGET_VALUE	GOAL_TYPE Domain values: <ul style="list-style-type: none"> <li>• OBJECTIVE</li> <li>• GOAL</li> <li>• POLICY</li> <li>• STRATEGY</li> <li>• MISSION</li> </ul> PRIORITY Domain values: <ul style="list-style-type: none"> <li>• MEDIUM</li> <li>• LOW</li> <li>• HIGH</li> </ul> RISK Domain values: <ul style="list-style-type: none"> <li>• MEDIUM</li> <li>• LOW</li> <li>• HIGH</li> </ul> CONFIDENCE_LEVEL Properties <ul style="list-style-type: none"> <li>• LOW</li> <li>• MEDIUM</li> <li>• HIGH</li> </ul>
<a href="#">HELP</a>	HELP_DESCRIPTION FORMAT NAME SYSID COUNTRY_LANGUAGE SOURCE_FILE	FORMAT Domain values: <ul style="list-style-type: none"> <li>• IPF</li> <li>• HPS</li> <li>• RTF</li> </ul>
<a href="#">HELP_TEXT</a>	COUNTRY_LANGUAGE	COUNTRY_LANGUAGE Domain values: <ul style="list-style-type: none"> <li>• US_ENGLISH</li> <li>• UK_ENGLISH</li> </ul>

<a href="#">IDENTIFIER</a>	NAME SYSID TYPE	TYPE Domain values: <ul style="list-style-type: none"> <li>• ALTERNATE</li> <li>• CANDIDATE</li> <li>• PRIMARY</li> </ul>
<a href="#">INFORMATION_NEED</a>	INFORMATION_NEED_DESCRIPTION PRIORITY RISK ACTUAL_START_DATE PLANNED_START_DATE	PRIORITY Domain values: <ul style="list-style-type: none"> <li>• MEDIUM</li> <li>• LOW</li> <li>• HIGH</li> </ul> RISK Domain values: <ul style="list-style-type: none"> <li>• MEDIUM</li> <li>• LOW</li> <li>• HIGH</li> </ul>
<a href="#">KEY</a>	IMPLEMENTATION_NAME NAME SYSID TYPE DELETE_RULE UNIQUE	TYPE Domain values: <ul style="list-style-type: none"> <li>• FOREIGN</li> <li>• INDEX</li> <li>• PRIMARY</li> </ul> DELETE_RULE Domain values: <ul style="list-style-type: none"> <li>• CASCADE</li> <li>• NULL</li> <li>• RESTRICT</li> </ul> UNIQUE Domain values: <ul style="list-style-type: none"> <li>• FALSE</li> <li>• TRUE</li> </ul>
<a href="#">LANGUAGE</a>	ABBREV	
<a href="#">LOCATION</a>	LOCATION_DESCRIPTION NAME SYSID	
<a href="#">LOGICAL_PROCESS</a>	LOGICAL_PROCESS_DESCRIPTION MODE NAME SYSID TYPE	MODE Domain values: <ul style="list-style-type: none"> <li>• ACTION</li> <li>• DECISION</li> </ul> TYPE Domain values: <ul style="list-style-type: none"> <li>• ASSOCIATE</li> <li>• CALCULATE</li> <li>• CAPTURE</li> <li>• DELETE</li> <li>• RETRIEVE</li> <li>• UPDATE</li> <li>• VALIDATE</li> <li>• VALIDATE</li> </ul>
<a href="#">MACHINE</a>	GROUP IMPLEMENTATION_NAME NAME OPERATING_SYSTEM OPERATING_SYSTEM_RELEASE SYSID	OPERATING_SYSTEM Domain values: <ul style="list-style-type: none"> <li>• AIX</li> <li>• CICS_MVS</li> <li>• HPUX</li> <li>• IMS</li> <li>• MVS</li> <li>• SUN_OS</li> <li>• WINDOWS</li> </ul>

<a href="#">MIGRATION</a>	STATE DSN RETURN_CODE SOURCE_FILE	MIGRATION_STATE Domain values: <ul style="list-style-type: none"> <li>• IMPORT_JOB_SUBMITTED</li> <li>• NO_ACTION_EXECUTED</li> <li>• IMPORT_JOB_FAILED</li> <li>• IMPORT_JOB_SUCCESSFUL</li> <li>• IMPORT_JOB_APPROVED</li> <li>• LOAD_JOB_FAILED</li> <li>• IMPORT_JOB_EXECUTING</li> <li>• EXPORT_JOB_EXECUTING</li> <li>• LOAD_JOB_SUCCESSFUL</li> <li>• EXPORT_JOB_FAILED</li> <li>• LOAD_JOB_APPROVED</li> <li>• EXPORT_JOB_SUCCESSFUL</li> <li>• ANALYZE_JOB_SUBMITTED</li> <li>• EXPORT_JOB_APPROVED</li> <li>• ANALYZE_JOB_EXECUTING</li> <li>• LOAD_JOB_SUBMITTED</li> <li>• ANALYZE_JOB_FAILED</li> <li>• LOAD_JOB_EXECUTING</li> <li>• ANALYZE_JOB_SUCCESSFUL</li> <li>• EXPORT_JOB_SUBMITTED</li> <li>• ANALYZE_JOB_APPROVED</li> </ul>
<a href="#">OPPORTUNITY</a>	OPPORTUNITY_DESCRIPTION BENEFIT EXPLT_COST IGNORE_COST ACTUAL_START_DATE PLANNED_START_DATE PRIORITY CONFIDENCE_LEVEL RISK STATUS TYPE OPERATING_SYSTEM	PRIORITY Domain values: <ul style="list-style-type: none"> <li>• MEDIUM</li> <li>• LOW</li> <li>• HIGH</li> </ul> CONFIDENCE_LEVEL Properties <ul style="list-style-type: none"> <li>• LOW</li> <li>• MEDIUM</li> <li>• HIGH</li> </ul> RISK Domain values: <ul style="list-style-type: none"> <li>• MEDIUM</li> <li>• LOW</li> <li>• HIGH</li> </ul> STATUS Domain values: <ul style="list-style-type: none"> <li>• IMPORT_JOB_SUBMITTED</li> <li>• FUNDED</li> <li>• REJECTED</li> <li>• SIZED</li> <li>• DEFERRED</li> </ul> OPERATING_SYSTEM Domain values: <ul style="list-style-type: none"> <li>• AIX</li> <li>• CICS_6000</li> <li>• CICS_MVS</li> <li>• HPUX</li> <li>• MVS</li> <li>• SUN_OS</li> <li>• WINDOWS</li> </ul>
<a href="#">ORGANIZATION</a>	NAME SYSID	

<p><a href="#">PANEL</a></p>	<p>CODEPAGE  BASE?  COUNTRY_LANGUAGE  GUI  SOURCE_FILE  COORDINATE_METHOD  PANEL_DESCRIPTION  NAME  SYSID  X_RESOLUTION  Y_RESOLUTION</p>	<p>BASE Domain values:</p> <ul style="list-style-type: none"> <li>• FALSE</li> <li>• TRUE</li> </ul> <p>COUNTRY_LANGUAGE Domain values:</p> <ul style="list-style-type: none"> <li>• US_ENGLISH</li> <li>• UK_ENGLISH</li> </ul> <p>GUI Domain values:</p> <ul style="list-style-type: none"> <li>• 3270</li> <li>• PWS_GENERIC</li> <li>• OPEN_LOOK</li> <li>• WINDOWS</li> </ul> <p>COORDINATE_METHOD Domain values:</p> <ul style="list-style-type: none"> <li>• CHARACTER</li> <li>• PIXEL</li> </ul>
<p><a href="#">PARTITION</a></p>	<p>CELL_RANK  CLIENT_TYPE  COLLECTION_ID  GENERAL_LANGUAGE  IMPLEMENTATION_PACKAGE  ISOLATION_MODE  LINK_TYPE  MAXIMUM_TRANS_ID  MINIMUM_TRANS_ID  NAME  PARTITION_TYPE  SERVER_INTERFACE  SERVER_OWNER  PLAN_NAME  QUALIFIER  SYSID</p>	<p>CLIENT_TYPE values:</p> <ul style="list-style-type: none"> <li>• HTML</li> <li>• CONVERSE</li> <li>• EVENT-DRIVEN</li> </ul> <p>GENERAL_LANGUAGE values:</p> <ul style="list-style-type: none"> <li>• DEFAULT</li> <li>• JAVA</li> </ul> <p>LINK_TYPE values:</p> <ul style="list-style-type: none"> <li>• DYNAMIC</li> <li>• STATIC</li> </ul> <p>SERVER_INTERFACE values:</p> <ul style="list-style-type: none"> <li>• AppBuilder Communications</li> <li>• RMI</li> <li>• ENTERPRISE_JAVABEANS</li> <li>• MQ-SERIES</li> </ul>
<p>PHYSICAL_EVENT</p>	<p>PHYSICAL_EVENT_DESCRIPTION  TYPE  EVENT_CLASS  EVENT_SCOPE</p>	<p>EVENT_CLASS domain values:</p> <ul style="list-style-type: none"> <li>• TEMPORAL</li> <li>• EXTERNAL</li> <li>• INTERNAL</li> </ul> <p>EVENT_SCOPE domain values:</p> <ul style="list-style-type: none"> <li>• GLOBAL</li> <li>• LOCAL</li> <li>• CELL</li> </ul>

<a href="#">PROCESS</a>	CHILD_MENU EXECUTION_ENVIRONMENT MENU_DESCRIPTION NAME SYSID WORKSTATION_GROUP	CHILD_MENU Domain values: <ul style="list-style-type: none"> <li>• MENU_BAR</li> <li>• NO_MENU</li> <li>• PULL_DOWN</li> </ul> EXECUTION_ENVIRONMENT Domain values: <ul style="list-style-type: none"> <li>• CICS</li> <li>• IMS</li> <li>• NOT_APPLICABLE</li> <li>• PC</li> <li>• PC_AND_CICS</li> <li>• PC_AND_IMS</li> </ul>
<a href="#">RELATIONSHIP</a>	ACTIVITY_PERIOD AVERAGE_DELETES AVERAGE_INSERTS AVERAGE_UPDATES EXPECTED_ROWS MAXIMUM_ROWS MINIMUM_ROWS NAME SYSID TYPE	ACTIVITY_PERIOD Domain values: <ul style="list-style-type: none"> <li>• DAY</li> <li>• MONTH</li> <li>• QUARTER</li> <li>• WEEK</li> <li>• YEAR</li> </ul> TYPE Domain values: <ul style="list-style-type: none"> <li>• AND</li> <li>• IOR</li> <li>• REGULAR</li> <li>• SUBTYPE</li> <li>• XOR</li> <li>• XOR</li> </ul>
<a href="#">REPORT</a>	EXECUTION_ENVIRONMENT IMPLEMENTATION_NAME LEFT_MARGIN LINE_SIZE NAME ORIENTATION PAGE_SIZE PRINTER_TYPE SYSID TOP_MARGIN	EXECUTION_ENVIRONMENT Domain values: <ul style="list-style-type: none"> <li>• BATCH</li> <li>• CICS</li> <li>• CICS_BATCH</li> </ul> ORIENTATION Domain values: <ul style="list-style-type: none"> <li>• LANDSCAPE</li> <li>• PORTRAIT</li> </ul> PRINTER_TYPE Domain values: <ul style="list-style-type: none"> <li>• 3800</li> <li>• GENERIC</li> </ul>

<a href="#">REQUIREMENT</a>	REQUIREMENT_DESCRIPTION TYPE CONFIDENCE_LEVEL PLANNED_START_DATE PRIORITY RISK	REQUIREMENT_TYPE Domain values: <ul style="list-style-type: none"> <li>• CUSTOMER</li> <li>• TECHNICAL</li> <li>• COMPETITION</li> <li>• BUSINESS</li> </ul> CONFIDENCE_LEVEL Properties <ul style="list-style-type: none"> <li>• LOW</li> <li>• MEDIUM</li> <li>• HIGH</li> </ul> PRIORITY Domain values: <ul style="list-style-type: none"> <li>• MEDIUM</li> <li>• LOW</li> <li>• HIGH</li> </ul> RISK Domain values: <ul style="list-style-type: none"> <li>• MEDIUM</li> <li>• LOW</li> <li>• HIGH</li> </ul>
<a href="#">RULE</a>	DBMS ENVIRONMENT IMPLEMENTATION_NAME ISOLATION_MODE NAME PACKAGE PLAN_NAME SOURCE_FILE SYSID TRANID	DBMS Domain values: <ul style="list-style-type: none"> <li>• DB2</li> <li>• N/A</li> </ul> ENVIRONMENT Domain values: <ul style="list-style-type: none"> <li>• CICS</li> <li>• CICS_AND_BATCH</li> <li>• IMS</li> <li>• LANDP_FUNCTION</li> <li>• LANDP_SERVER</li> <li>• MVS_BATCH</li> <li>• PC</li> <li>• PC_AND_CICS</li> <li>• PC_AND_IMS</li> <li>• STRATUS</li> </ul> EXECUTION_MODE Domain values: <ul style="list-style-type: none"> <li>• ASYNCHRONOUS</li> <li>• SYNCHRONOUS</li> </ul>
<a href="#">SECTION</a>	IMPLEMENTATION_NAME SOURCE_FILE NAME SYSID	
<a href="#">SERVER</a>	COLLECTION_ID IMPLEMENTATION_NAME ISOLATION_MODE MAXIMUM_TRANSACTION_ID MINIMUM_TRANSACTION_ID NAME NEXT_TRANSACTION_ID	ISOLATION_MODE Domain values: <ul style="list-style-type: none"> <li>• GET</li> <li>• SET</li> <li>• XDR-GET</li> <li>• XDR-SET</li> </ul>

<a href="#">SET</a>	FRACTION IMPLEMENTATION_NAME LENGTH NAME PICTURE REPRESENTATION_LENGTH STYLE SYSID TYPE	STYLE Domain values: <ul style="list-style-type: none"> <li>• DEFINE</li> <li>• ERROR</li> <li>• LOOKUP</li> <li>• VALUES</li> </ul> TYPE Domain values: <ul style="list-style-type: none"> <li>• CHARACTER</li> <li>• DECIMAL</li> <li>• GRAPHIC_CHAR</li> <li>• INTEGER</li> <li>• MIXED_CHAR</li> </ul>
<a href="#">SOFTWARE_PACKAGE</a>	STATE VALIDATE_FLAG	VALIDATE_FLAG Domain values: <ul style="list-style-type: none"> <li>• FALSE</li> <li>• TRUE</li> </ul>
<a href="#">STATE</a>	STATE_DESCRIPTION NAME SYSID TYPE	TYPE Domain values: <ul style="list-style-type: none"> <li>• FINAL</li> <li>• INITIAL</li> <li>• INTERMEDIATE</li> </ul>
<a href="#">SUCCESS_FACTOR</a>	SUCCESS_FACTOR_DESCRIPTION CONFIDENCE_LEVEL PRIORITY ACTUAL_START_DATE PLANNED_START_DATE RISK TARGET_VALUE	CONFIDENCE_LEVEL Properties: <ul style="list-style-type: none"> <li>• LOW</li> <li>• MEDIUM</li> <li>• HIGH</li> </ul> PRIORITY Domain values: <ul style="list-style-type: none"> <li>• MEDIUM</li> <li>• LOW</li> <li>• HIGH</li> </ul> RISK Domain values: <ul style="list-style-type: none"> <li>• MEDIUM</li> <li>• LOW</li> <li>• HIGH</li> <li>• HIGH</li> </ul>
<a href="#">SYMBOL</a>	DEFINE ENCODING DISPLAY	
<a href="#">SYSTEM</a>	SYSTEM_DESCRIPTION NAME SYSID	
<a href="#">TABLE</a>	ACTIVATION_PERIOD AVE_DELETES AVE_INSERTS AVE_UPDATES CREATOR IMPLEMENTATION_NAME MAXIMUM_ROWS MINIMUM_ROWS NAME SYSID TYPE	ACTIVATION_PERIOD Domain values: <ul style="list-style-type: none"> <li>• DAY</li> <li>• MONTH</li> <li>• QUARTER</li> <li>• WEEK</li> <li>• YEAR</li> </ul> TYPE Domain values: <ul style="list-style-type: none"> <li>• TABLE</li> <li>• VIEW</li> </ul>
<a href="#">TRANSITION</a>	NAME SYSID	

<a href="#">VALUE</a>	IMPLEMENTATION_NAME NAME SYSID	
<a href="#">VERSION</a>	STATE VERSION_DESCRIPTION	<ul style="list-style-type: none"> <li>STATE Domain values:</li> <li>ROOT</li> <li>DELETED</li> <li>FIXED</li> <li>RELEASE</li> <li>WORKING</li> </ul>
<a href="#">VIEW</a>	IMPLEMENTATION_NAME VIEW_NAME SYSID	
<a href="#">WINDOW</a>	IMPLEMENTATION_NAME NAME SYSID	
<a href="#">WINDOW_CONTENT</a>	GUI NAME SYSID	GUI Domain values: <ul style="list-style-type: none"> <li>3270</li> <li>PWS_GENERIC</li> <li>OPEN_LOOK</li> <li>OSF_MOTIF</li> <li>WINDOWS</li> </ul>

## Entities and Relationships

The following table provides information on entities and relations in the TurboCycler Developer's Kit. TurboCycler is not supported for the new OO objects.

### Relationships

Parent	Relationship	Relationship Properties
<a href="#">APPLICATION_CONFIGURATION</a>	APPLICATION_CONFIGURATION_HAS_CONFIGURATION_UNIT	Sequence Number (1-999)
<a href="#">ASSUMPTION</a>	ASSUMPTION_AFFECTS	SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">ASSUMPTION</a>	ASSUMPTION_AFFECTS	SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">ASSUMPTION</a>	ASSUMPTION_SUPPORTED_BY	SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">ASSUMPTION</a>	ASSUMPTION_SUPPORTED_BY	SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT



<a href="#">ASSUMPTION</a>	ASSUMPTION_SUPPORTED_BY	SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">ASSUMPTION</a>	ASSUMPTION_SUPPORTED_BY	SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">ATTRIBUTE</a>	CONNECTS_TO	SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">ATTRIBUTE</a>	CONNECTS_TO	SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">ATTRIBUTE</a>	CONNECTS_TO	SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">ATTRIBUTE</a>	CONNECTS_TO	SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">ATTRIBUTE</a>	IMPLEMENTED_BY	SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">ATTRIBUTE</a>	IMPLEMENTED_BY	SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">ATTRIBUTE</a>	IMPLEMENTED_BY	SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">ATTRIBUTE</a>	IMPLEMENTED_BY	SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">ATTRIBUTE</a>	IMPLEMENTED_BY	SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">ATTRIBUTE</a>	IS_COMPOSED_OF_ATTRIBUTE	SEQUENCE NUMBER (1-999) OCCURS OBJECT_TEXT
<a href="#">ATTRIBUTE</a>	IS_TYPED_BY	SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">BITMAP</a>	BITMAP_HAS_BITMAP_IMPLEMENTATION	IS_COMPOSED_OF_ATTRIBUTE P SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">BITMAP_IMPLEMENTATION</a>	PHYSICAL_BITMAP_FILE	
<a href="#">BUSINESS_OBJECT</a>	BUSINESS_OBJECT_OWNS	BUSINESS_OBJECT_OWNS Proper SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">BUSINESS_OBJECT</a>	BUSINESS_OBJECT_OWNS	BUSINESS_OBJECT_OWNS Proper SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">BUSINESS_OBJECT</a>	BUSINESS_OBJECT_OWNS	BUSINESS_OBJECT_OWNS Proper SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">BUSINESS_OBJECT</a>	BUSINESS_OBJECT_REFINES_INTO_BUSINESS_OBJECT	BUSINESS_OBJECT_REFINES_INT Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">BUSINESS_OBJECT</a>	HAS_STATE	HAS_STATE Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT

<a href="#">BUSINESS_OBJECT</a>	HAS_STD_DRAWING	HAS_STD_DRAWING Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">COLLECTION</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">COLLECTION</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">COLLECTION</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">COLLECTION</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">COLLECTION</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">COLLECTION</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">COLLECTION</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">COLLECTION</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">COLLECTION</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">COLUMN</a>	[None]	
<a href="#">COMPONENT</a>	OWNS_VIEW	OWNS_VIEW Properties: SEQUENCE_NUMBER VIEW_USAGE OBJECT_TEXT
<a href="#">COMPONENT</a>	REFERS_TO	REFERS_TO Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">COMPONENT</a>	ACCESSES	ACCESSES Properties: SEQUENCE_NUMBER COMP_USAGE OBJECT_TEXT
<a href="#">COMPONENT</a>	COMPONENT_USES_COMPONENT	COMPONENT_USES_COMPONENT SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">COMPONENT</a>	COMPONENT_SOURCE	[None]
<a href="#">CONTEXT_DIAGRAM</a>	CONTEXT_DIAGRAM_CONTAINS	CONTEXT_DIAGRAM_CONTAINS F SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">CONTEXT_DIAGRAM</a>	CONTEXT_DIAGRAM_CONTAINS	CONTEXT_DIAGRAM_CONTAINS F SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">CONTEXT_DIAGRAM</a>	CONTEXT_DIAGRAM_CONTAINS	CONTEXT_DIAGRAM_CONTAINS F SEQUENCE_NUMBER OBJECT_TEXT

<a href="#">CONTEXT_DIAGRAM</a>	CONTEXT_DIAGRAM_CONTAINS	CONTEXT_DIAGRAM_CONTAINS F SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">CONTEXT_DIAGRAM</a>	CONTEXT_DIAGRAM_CONTAINS	CONTEXT_DIAGRAM_CONTAINS F SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">CONTEXT_DIAGRAM</a>	DATA_CONTENT_DEFINED_BY	DATA_CONTENT_DEFINED_BY Prc SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">DATABASE</a>	ACCESSES	ACCESSES Properties: SEQUENCE_NUMBER COMP_USAGE OBJECT_TEXT
<a href="#">DATABASE</a>	DATABASE_HAS_TABLE	DATABASE_HAS_TABLE Properties SEQUENCE_NUMBER MINIMUM_ROWS MAXIMUM_ROWS ACTIVE_PERIOD AVE_INSERTS AVE_DELETES DONE_FLAG DO_FLAG OBJECT_TEXT
<a href="#">DATABASE</a>	DATABASE_RELATED_TO_DATABASE	DATABASE_RELATED_TO_DATAB. SEQUENCE_NUMBER REL_TYPE SHADOWED_TABLE OBJECT_TEXT
<a href="#">DATA_FLOW</a>	DATA_CONTENT_DEFINED_BY	DATA_CONTENT_DEFINED_BY Prc SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">DATA_STORE</a>	DATA_CONTENT_DEFINED_BY	DATA_CONTENT_DEFINED_BY Prc SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">DATA_STORE</a>	DATA_STORE_IS_REPLACED_BY_ENTITY	DATA_STORE_IS_REPLACED_BY_ SEQUENCE_NUMBER COMMENTS OBJECT_TEXT
<a href="#">DATA_TYPE</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">DATA_TYPE</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">DATA_TYPE</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">DATA_TYPE</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">DATA_TYPE</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">DATA_TYPE</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">DATA_TYPE</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT

<a href="#">DATA_TYPE</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">DATA_TYPE</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">DATA_TYPE</a>	DATATYPE_IS_CONSTRAINED_BY_SET	DATATYPE_IS_CONSTRAINED_BY SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">DATA_TYPE</a>	DATATYPE_IS_COMPOSED_OF_DATATYPE	DATATYPE_IS_COMPOSED_OF_D SEQUENCE_NUMBER OCCURS OBJECT_TEXT
<a href="#">DEVELOPMENT_PROJECT</a>	DEV_PROJECT_ADDRESSES	DEV_PROJECT_ADDRESSES Pro SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">DEVELOPMENT_PROJECT</a>	DEV_PROJECT_ADDRESSES	DEV_PROJECT_ADDRESSES Pro SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">DEVELOPMENT_PROJECT</a>	DEV_PROJECT_ADDRESSES	DEV_PROJECT_ADDRESSES Pro SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">DEVELOPMENT_PROJECT</a>	DEV_PROJECT_CONTAINS	DEV_PROJECT_CONTAINS Propert SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">DEVELOPMENT_PROJECT</a>	DEV_PROJECT_INCLUDES	DEV_PROJECT_INCLUDES Propert SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">DEVELOPMENT_PROJECT</a>	DEV_PROJECT_INCLUDES	DEV_PROJECT_INCLUDES Propert SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">DRAWING</a>	[none]	
<a href="#">ENTITY</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">ENTITY</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">ENTITY</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">ENTITY</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">ENTITY</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">ENTITY</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">ENTITY</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">ENTITY</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT

<a href="#">ENTITY</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">ENTITY</a>	ENTITY_IS_DISTRIBUTED_AT	ENTITY_IS_DISTRIBUTED_AT Prop SEQUENCE_NUMBER MASTER VARIANT PARTITIONED REPLICATED SUBSET REORGANIZED TELEPROC COMMENTS OBJECT_TEXT
<a href="#">ENTITY</a>	HAS_IDENTIFIER	HAS_IDENTIFIER Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">ENTITY</a>	HAS_STATE	HAS_STATE Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">ENTITY</a>	OWNS_VIEW	OWNS_VIEW Properties: SEQUENCE_NUMBER VIEW_USAGE OBJECT_TEXT
<a href="#">ENTITY</a>	IS_DESCRIBED_BY	IS_DESCRIBED_BY Properties: SEQUENCE_NUMBER OPTIONALITY MINIMUM_PER_SUBJECT MAXIMUM_PER_SUBJECT OBJECT_TEXT
<a href="#">ENTITY</a>	IS_RELATED_VIA	IS_RELATED_VIA Properties: SEQUENCE_NUMBER CONTROLLING CARDINALITY OPTIONALITY DEPENDENT ABSTRACT ROLE MINIMUM_CARDINALITY MAXIMUM_CARDINALITY OBJECT_TEXT
<a href="#">ENTITY</a>	ENTITY_IS_MODIFIED_AT_LOCATION	ENTITY_IS_MODIFIED_AT_LOCATION SEQUENCE_NUMBER CREATE READ UPDATES DELETE COMMENTS OBJECT_TEXT
<a href="#">ENTITY</a>	ENTITY_MODIFIED_BY	ENTITY_MODIFIED_BY Properties: SEQUENCE_NUMBER CREATES READS UPDATES DELETES COMMENTS OBJECT_TEXT
<a href="#">ENTITY</a>	ENTITY_MODIFIED_BY	ENTITY_MODIFIED_BY Properties: SEQUENCE_NUMBER CREATES READS UPDATES DELETES COMMENTS OBJECT_TEXT

<a href="#">ENTITY</a>	ENTITY_MODIFIED_BY	ENTITY_MODIFIED_BY Properties: SEQUENCE_NUMBER CREATES READS UPDATES DELETES COMMENTS OBJECT_TEXT
<a href="#">ENTITY</a>	ENTITY_MODIFIED_BY	ENTITY_MODIFIED_BY Properties: SEQUENCE_NUMBER CREATES READS UPDATES DELETES COMMENTS OBJECT_TEXT
<a href="#">ENTITY</a>	ENTITY_MODIFIED_BY	ENTITY_MODIFIED_BY Properties: SEQUENCE_NUMBER CREATES READS UPDATES DELETES COMMENTS OBJECT_TEXT
<a href="#">ENTITY</a>	HAS_ERD_DRAWING	HAS_ERD_DRAWING Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">ENTITY</a>	HAS_STD_DRAWING	HAS_STD_DRAWING Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">ER_VIEW</a>	ER_VIEW_INVOLVES	ER_VIEW_INVOLVES Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">ER_VIEW</a>	ER_VIEW_INVOLVES	ER_VIEW_INVOLVES Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">ER_VIEW</a>	ER_VIEW_INVOLVES	ER_VIEW_INVOLVES Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">ER_VIEW</a>	ER_VIEW_INVOLVES	ER_VIEW_INVOLVES Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">ER_VIEW</a>	ER_VIEW_CONTAINS	ER_VIEW_CONTAINS Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">EVENT</a>	DATA_CONTENT_DEFINED_BY	DATA_CONTENT_DEFINED_BY Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">EVENT</a>	HAS_PDD_DRAWING	HAS_PDD_DRAWING Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">EVENT</a>	IS_COMPOSED_OF_EVENT	IS_COMPOSED_OF_EVENT Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">EVENT</a>	EVENT_CAUSES	EVENT_CAUSES Properties: SEQUENCE_NUMBER EVENT_CAUSES_DESCRIPTION OBJECT_TEXT

<a href="#">EVENT</a>	EVENT_TRIGGERS	EVENT_TRIGGERS Properties: SEQUENCE_NUMBER EVENT_TRIGGERS_DESCRIPTION INCLUSIVE_FLAG EXOR_SEQUENCE_NUMBER
<a href="#">EVENT</a>	TRIGGERS_LOGICAL_PROCESS	TRIGGERS_LOGICAL_PROCESS P SEQUENCE_NUMBER TRIGGERS_LOGICAL_PROCESS_I OBJECT_TEXT
<a href="#">EVENT</a>	EVENT_INFLUENCES_BUSINESS_OBJECT	EVENT_INFLUENCES_BUSINESS_ SEQUENCE_NUMBER EVENT_INFLUENCES_BUSINESS_  OBJECT_TEXT
<a href="#">EXTERNAL_AGENT</a>	DATA_CONTENT_DEFINED_BY	DATA_CONTENT_DEFINED_BY Pr SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">EXTERNAL_AGENT</a>	INITIATES_EVENT	INITIATES_EVENT Properties: SEQUENCE_NUMBER INITIATES_EVENT_DESCRIPTION OBJECT_TEXT
<a href="#">FIELD</a>	REFERS_TO	REFERS_TO Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">FIELD</a>	USES_LANGUAGE	USES_LANGUAGE Properties: SEQUENCE_NUMBER VALUE_SHORT VALUE_LONG OBJECT_TEXT
<a href="#">FIELD</a>	HAS_HELP_TEXT	HAS_HELP_TEXT Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">FILE</a>	FILE_IS_FORWARDED_TO_FILE	FILE_IS_FORWARDED_TO_FILE P SEQUENCE_NUMBER UPDATE_RULE KEY_DELETE_RULE INSERT_RULE OBJECT_TEXT
<a href="#">FILE</a>	IS_KEYED_BY	IS_KEYED_BY Properties: SEQUENCE_NUMBER TYPE OBJECT_TEXT
<a href="#">FILE</a>	OWNS_VIEW	OWNS_VIEW Properties: SEQUENCE_NUMBER VIEW_USAGE OBJECT_TEXT
<a href="#">FORM</a>	[none]	
<a href="#">FSDM_SCHEMA</a>	FSDM_SCHEMA_HAS_FSDM_VALUE	FSDM_SCHEMA_HAS_FSDM_VALL SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">FSDM_VALUE</a>	FSDM_VALUE_CLASSIFIES_FSDM_SCHEMA	FSDM_VALUE_CLASSIFIES_FSDM SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">FUNCTION</a>	HAS_BITMAP	HAS_BITMAP Properties: SEQUENCE_NUMBER OBJECT_TEXT

<a href="#">FUNCTION</a>	FUNCTION_INTERSECTS_WITH_ENTITY	FUNCTION_INTERSECTS_WITH_E SEQUENCE_NUMBER CREATES READS UPDATES DELETES COMMENTS OBJECT_TEXT
<a href="#">FUNCTION</a>	FUNCTION_IS_CARRIED_OUT_AT_LOCATION	FUNCTION_IS_CARRIED_OUT_AT_ SEQUENCE_NUMBER MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">FUNCTION</a>	REFINES_INTO	REFINES_INTO Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">FUNCTION</a>	USES_LANGUAGE	USES_LANGUAGE Properties: SEQUENCE_NUMBER VALUE_SHORT VALUE_LONG OBJECT_TEXT
<a href="#">GOAL</a>	GOAL_CONTAINS	GOAL_CONTAINS Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">GOAL</a>	GOAL_INVOLVES	FUNCTION_INTERSECTS_WITH_E SEQUENCE_NUMBER CREATES READS UPDATES DELETES COMMENTS OBJECT_TEXT
<a href="#">GOAL</a>	GOAL_SUPPORTED_BY	GOAL_SUPPORTED_BY Properties: SEQUENCE_NUMBER CREATES READS UPDATES DELETES COMMENTS OBJECT_TEXT
<a href="#">GOAL</a>	GOAL_SUPPORTED_BY	GOAL_SUPPORTED_BY Properties: SEQUENCE_NUMBER CREATES READS UPDATES DELETES COMMENTS OBJECT_TEXT
<a href="#">GOAL</a>	GOAL_SUPPORTED_BY	GOAL_SUPPORTED_BY Properties: SEQUENCE_NUMBER CREATES READS UPDATES DELETES COMMENTS OBJECT_TEXT
<a href="#">GOAL</a>	GOAL_SUPPORTED_BY	GOAL_SUPPORTED_BY Properties: SEQUENCE_NUMBER CREATES READS UPDATES DELETES COMMENTS OBJECT_TEXT



<a href="#">GOAL</a>	GOAL_SUPPORTED_BY	GOAL_SUPPORTED_BY Properties: SEQUENCE_NUMBER CREATES READS UPDATES DELETES COMMENTS OBJECT_TEXT
<a href="#">GOAL</a>	GOAL_SUPPORTED_BY	GOAL_SUPPORTED_BY Properties: SEQUENCE_NUMBER CREATES READS UPDATES DELETES COMMENTS OBJECT_TEXT
<a href="#">HELP</a>	[none]	
<a href="#">HELP_TEXT</a>	[none]	
<a href="#">IDENTIFIER</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">IDENTIFIER</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">IDENTIFIER</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">IDENTIFIER</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">IDENTIFIER</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">IDENTIFIER</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">IDENTIFIER</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">IDENTIFIER</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">IDENTIFIER</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">IDENTIFIER</a>	IDENTIFIER_IS_COMPOSED_OF	IDENTIFIER_IS_COMPOSED_OF P SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">IDENTIFIER</a>	IDENTIFIER_IS_COMPOSED_OF	IDENTIFIER_IS_COMPOSED_OF P SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">INFORMATION_NEED</a>	INFO_NEED_CONTAINS	INFO_NEED_CONTAINS Properties: SEQUENCE_NUMBER OBJECT_TEXT

<a href="#">INFORMATION_NEED</a>	INFO_NEED_AFFECTS	INFO_NEED_AFFECTS Properties: SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">INFORMATION_NEED</a>	INFO_NEED_AFFECTS	INFO_NEED_AFFECTS Properties: SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">INFORMATION_NEED</a>	INFO_NEED_SUPPORTED_BY	INFO_NEED_SUPPORTED_BY Prop SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">INFORMATION_NEED</a>	INFO_NEED_SUPPORTED_BY	INFO_NEED_SUPPORTED_BY Prop SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">INFORMATION_NEED</a>	INFO_NEED_SUPPORTED_BY	INFO_NEED_SUPPORTED_BY Prop SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">INFORMATION_NEED</a>	INFO_NEED_AFFECTED_BY	INFO_NEED_AFFECTED_BY Prop SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">KEY</a>	KEY_HAS_COLUMN	KEY_HAS_COLUMN Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">LANGUAGE</a>	[none]	
<a href="#">LOCATION</a>	LOCATION_IS_SITE_OF	LOCATION_IS_SITE_OF Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">LOCATION</a>	LOCATION_IS_SITE_OF	LOCATION_IS_SITE_OF Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">LOCATION</a>	LOCATION_IS_SITE_OF	LOCATION_IS_SITE_OF Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">LOCATION</a>	LOCATION_USES	LOCATION_USES Properties: SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT

<a href="#">LOCATION</a>	LOCATION_USES	LOCATION_USES Properties: SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">LOCATION</a>	LOCATION_USES	LOCATION_USES Properties: SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">LOCATION</a>	LOCATION_USES	LOCATION_USES Properties: SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">LOCATION</a>	LOCATION_CONTAINS	LOCATION_CONTAINS Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">LOGICAL_PROCESS</a>	DATA_CONTENT_DEFINED_BY	DATA_CONTENT_DEFINED_BY Proc SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">LOGICAL_PROCESS</a>	INITIATES_EVENT	INITIATES_EVENT Properties: SEQUENCE_NUMBER INITIATES_EVENT_DESCRIPTION OBJECT_TEXT
<a href="#">LOGICAL_PROCESS</a>	HAS_PDD_DRAWING	HAS_PDD_DRAWING Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">LOGICAL_PROCESS</a>	LOGICAL_PROCESS_AFFECTS	LOGICAL_PROCESS_AFFECTS Proc SEQUENCE_NUMBER CREATES READS UPDATES DELETES COMMENTS OBJECT_TEXT
<a href="#">LOGICAL_PROCESS</a>	DEPENDS_ON_LOGICAL_PROCESS	DEPENDS_ON_LOGICAL_PROCESS SEQUENCE_NUMBER DEPENDS_ON_LOGICAL_PROCESS OBJECT_TEXT
<a href="#">LOGICAL_PROCESS</a>	IS_COMPOSED_OF_LOGICAL_PROCESS	IS_COMPOSED_OF_LOGICAL_PROCESS SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">LOGICAL_PROCESS</a>	LOGICAL_PROCESS_IMPLEMENTED_BY	LOGICAL_PROCESS_IMPLEMENTED_BY SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">LOGICAL_PROCESS</a>	LOGICAL_PROCESS_CONTAINS	LOGICAL_PROCESS_CONTAINS P SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">LOGICAL_PROCESS</a>	LOGICAL_PROCESS_CONTAINS	LOGICAL_PROCESS_CONTAINS P SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">LOGICAL_PROCESS</a>	LOGICAL_PROCESS_CONTAINS	LOGICAL_PROCESS_CONTAINS P SEQUENCE_NUMBER OBJECT_TEXT

<a href="#">LOGICAL_PROCESS</a>	LOGICAL_PROCESS_SIGNED_BY	LOGICAL_PROCESS_SIGNATURE SEQUENCE_NUMBER SIGNATURE_TYPE OBJECT_TEXT
<a href="#">LOGICAL_PROCESS</a>	LOGICAL_PROCESS_SUPPORTS	LOGICAL_PROCESS_SUPPORTS F SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">LOGICAL_PROCESS</a>	LOGICAL_PROCESS_SUPPORTS	LOGICAL_PROCESS_SUPPORTS F SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">LOGICAL_PROCESS</a>	LOGICAL_PROCESS_SUPPORTED_BY	LOGICAL_PROCESS_SUPPORTED SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">LOGICAL_PROCESS</a>	LOGICAL_PROCESS_SUPPORTED_BY	LOGICAL_PROCESS_SUPPORTED SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">MACHINE</a>	MACHINE_CAN_ACCESS_MACHINE	MACHINE_CAN_ACCESS_MACHIN SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS











<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">MIGRATION</a>	MIGRATION_COMPRISED_OF	MIGRATION_COMPRISED_OF Prop SEQUENCE_NUMBER MIGRATION_COMPRISED_OF_DES SCOPE_TYPE SEED_STATUS
<a href="#">OPPORTUNITY</a>	OPPORTUNITY_CONTAINS	OPPORTUNITY_CONTAINS Propert SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">ORGANIZATION</a>	ORGANIZATION_SUPPORTS_FUNCTION	ORGANIZATION_SUPPORTS_FUNI SEQUENCE_NUMBER RESPONSIBLE MAJOR_INVOLVEMENT MINOR_INVOLVEMENT COMMENTS OBJECT_TEXT
<a href="#">ORGANIZATION</a>	ORGANIZATION_SUPPORTS_PROCESS	ORGANIZATION_SUPPORTS_PRO SEQUENCE_NUMBER RESPONSIBLE MAJOR_INVOLVEMENT MINOR_INVOLVEMENT COMMENTS OBJECT_TEXT



<a href="#">ORGANIZATION</a>	ORGANIZATION_CITES	ORGANIZATION_CITES Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">ORGANIZATION</a>	ORGANIZATION_CITES	ORGANIZATION_CITES Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">ORGANIZATION</a>	ORGANIZATION_CITES	ORGANIZATION_CITES Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">ORGANIZATION</a>	ORGANIZATION_RESPONSIBLE_FOR	ORGANIZATION_RESPONSIBLE_F SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">ORGANIZATION</a>	ORGANIZATION_RESPONSIBLE_FOR	ORGANIZATION_RESPONSIBLE_F SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">ORGANIZATION</a>	ORGANIZATION_RESPONSIBLE_FOR	ORGANIZATION_RESPONSIBLE_F SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">ORGANIZATION</a>	ORGANIZATION_RESPONSIBLE_FOR	ORGANIZATION_RESPONSIBLE_F SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">ORGANIZATION</a>	ORGANIZATION_RESPONSIBLE_FOR	ORGANIZATION_RESPONSIBLE_F SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">ORGANIZATION</a>	ORGANIZATION_RESPONSIBLE_FOR	ORGANIZATION_RESPONSIBLE_F SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">PANEL</a>	[none]	
<a href="#">PARTITION</a>	CONFIGURATION_UNIT_ENCAPSULATES	CONFIGURATION_UNIT_ENCAPSL SEQUENCE_NUMBER PREPARE_TIME IMPLEMENTATION_NAME SERVICE_NAME SIGNIFICANT_TIME LINK_TYPE SERVER_OWNER QUALIFIER PLAN_NAME COLLECTION_ID VERSION_ID ISOLATION_MODE OBJECT_TEXT
<a href="#">PARTITION</a>	CONFIGURATION_UNIT_ENCAPSULATES	CONFIGURATION_UNIT_ENCAPSL SEQUENCE_NUMBER PREPARE_TIME IMPLEMENTATION_NAME SERVICE_NAME SIGNIFICANT_TIME LINK_TYPE SERVER_OWNER QUALIFIER PLAN_NAME COLLECTION_ID VERSION_ID ISOLATION_MODE OBJECT_TEXT

<a href="#">PARTITION</a>	CONFIGURATION_UNIT_ENCAPSULATES	CONFIGURATION_UNIT_ENCAPSL SEQUENCE_NUMBER PREPARE_TIME IMPLEMENTATION_NAME SERVICE_NAME SIGNIFICANT_TIME LINK_TYPE SERVER_OWNER QUALIFIER PLAN_NAME COLLECTION_ID VERSION_ID ISOLATION_MODE OBJECT_TEXT
<a href="#">PARTITION</a>	CONFIGURATION_UNIT_ENCAPSULATES	CONFIGURATION_UNIT_ENCAPSL SEQUENCE_NUMBER PREPARE_TIME IMPLEMENTATION_NAME SERVICE_NAME SIGNIFICANT_TIME LINK_TYPE SERVER_OWNER QUALIFIER PLAN_NAME COLLECTION_ID VERSION_ID ISOLATION_MODE OBJECT_TEXT
<a href="#">PARTITION</a>	CONFIGURATION_UNIT_ENCAPSULATES	CONFIGURATION_UNIT_ENCAPSL SEQUENCE_NUMBER PREPARE_TIME IMPLEMENTATION_NAME SERVICE_NAME SIGNIFICANT_TIME LINK_TYPE SERVER_OWNER QUALIFIER PLAN_NAME COLLECTION_ID VERSION_ID ISOLATION_MODE OBJECT_TEXT
<a href="#">PARTITION</a>	CONFIGURATION_UNIT_ENCAPSULATES	CONFIGURATION_UNIT_ENCAPSL SEQUENCE_NUMBER PREPARE_TIME IMPLEMENTATION_NAME SERVICE_NAME SIGNIFICANT_TIME LINK_TYPE SERVER_OWNER QUALIFIER PLAN_NAME COLLECTION_ID VERSION_ID ISOLATION_MODE OBJECT_TEXT
<a href="#">PARTITION</a>	CONFIGURATION_UNIT_ENCAPSULATES	CONFIGURATION_UNIT_ENCAPSL SEQUENCE_NUMBER PREPARE_TIME IMPLEMENTATION_NAME SERVICE_NAME SIGNIFICANT_TIME LINK_TYPE SERVER_OWNER QUALIFIER PLAN_NAME COLLECTION_ID VERSION_ID ISOLATION_MODE OBJECT_TEXT
<a href="#">PARTITION</a>	CONFIGURATION_UNIT_ENCAPSULATES	CONFIGURATION_UNIT_ENCAPSL SEQUENCE_NUMBER PREPARE_TIME IMPLEMENTATION_NAME SERVICE_NAME SIGNIFICANT_TIME LINK_TYPE SERVER_OWNER QUALIFIER PLAN_NAME COLLECTION_ID VERSION_ID ISOLATION_MODE OBJECT_TEXT

<a href="#">PARTITION</a>	CONFIGURATION_UNIT_ENCAPSULATES	CONFIGURATION_UNIT_ENCAPSL SEQUENCE_NUMBER PREPARE_TIME IMPLEMENTATION_NAME SERVICE_NAME SIGNIFICANT_TIME LINK_TYPE SERVER_OWNER QUALIFIER PLAN_NAME COLLECTION_ID VERSION_ID ISOLATION_MODE OBJECT_TEXT
<a href="#">PARTITION</a>	CONFIGURATION_UNIT_ENCAPSULATES	CONFIGURATION_UNIT_ENCAPSL SEQUENCE_NUMBER PREPARE_TIME IMPLEMENTATION_NAME SERVICE_NAME SIGNIFICANT_TIME LINK_TYPE SERVER_OWNER QUALIFIER PLAN_NAME COLLECTION_ID VERSION_ID ISOLATION_MODE OBJECT_TEXT
<a href="#">PARTITION</a>	CONFIGURATION_UNIT_ENCAPSULATES	CONFIGURATION_UNIT_ENCAPSL SEQUENCE_NUMBER PREPARE_TIME IMPLEMENTATION_NAME SERVICE_NAME SIGNIFICANT_TIME LINK_TYPE SERVER_OWNER QUALIFIER PLAN_NAME COLLECTION_ID VERSION_ID ISOLATION_MODE OBJECT_TEXT
<a href="#">PARTITION</a>	CONFIGURATION_UNIT_ENCAPSULATES	CONFIGURATION_UNIT_ENCAPSL SEQUENCE_NUMBER PREPARE_TIME IMPLEMENTATION_NAME SERVICE_NAME SIGNIFICANT_TIME LINK_TYPE SERVER_OWNER QUALIFIER PLAN_NAME COLLECTION_ID VERSION_ID ISOLATION_MODE OBJECT_TEXT
<a href="#">PARTITION</a>	CONFIGURATION_UNIT_ENCAPSULATES	CONFIGURATION_UNIT_ENCAPSL SEQUENCE_NUMBER PREPARE_TIME IMPLEMENTATION_NAME SERVICE_NAME SIGNIFICANT_TIME LINK_TYPE SERVER_OWNER QUALIFIER PLAN_NAME COLLECTION_ID VERSION_ID ISOLATION_MODE OBJECT_TEXT
<a href="#">PARTITION</a>	CONFIGURATION_UNIT_ENCAPSULATES	CONFIGURATION_UNIT_ENCAPSL SEQUENCE_NUMBER PREPARE_TIME IMPLEMENTATION_NAME SERVICE_NAME SIGNIFICANT_TIME LINK_TYPE SERVER_OWNER QUALIFIER PLAN_NAME COLLECTION_ID VERSION_ID ISOLATION_MODE OBJECT_TEXT

<a href="#">PARTITION</a>	USES_LANGUAGE	USES_LANGUAGE Properties: SEQUENCE_NUMBER VALUE_SHORT VALUE_LONG OBJECT_TEXT
<a href="#">PROCESS</a>	PROCESS_DEPENDS_ON_PROCESS	PROCESS_DEPENDS_ON_PROCE SEQUENCE NUMBER (1-999) PROCESS_DEPENDS_ON_PROCE INCLUSIVE_FLAG EXOR_SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">PROCESS</a>	HAS_BITMAP	HAS_BITMAP Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">PROCESS</a>	IS_DEFINED_BY	IS_DEFINED_BY Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">PROCESS</a>	PROCESS_IS_CARRIED_OUT_AT_LOCATION	PROCESS_IS_CARRIED_OUT_AT_ SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">PROCESS</a>	PROCESS_REFINES_INTO_PROCESS	PROCESS_REFINES_INTO_PROCE SEQUENCE NUMBER (1-999) CONDITIONAL_FLAG OBJECT_TEXT
<a href="#">PROCESS</a>	USES_LANGUAGE	USES_LANGUAGE Properties: SEQUENCE_NUMBER VALUE_SHORT VALUE_LONG OBJECT_TEXT
<a href="#">PROCESS</a>	PROCESS_REPLACES_SYSTEM	PROCESS_REPLACES_SYSTEM P SEQUENCE NUMBER (1-999) CONDITIONAL_FLAG OBJECT_TEXT
<a href="#">PROCESS</a>	PROCESS_IMPACTS_ENTITY	PROCESS_IMPACTS_ENTITY Prop SEQUENCE NUMBER (1-999) CREATES READS UPDATES DELETES COMMENTS OBJECT_TEXT
<a href="#">RELATIONSHIP</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">RELATIONSHIP</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">RELATIONSHIP</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">RELATIONSHIP</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">RELATIONSHIP</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">RELATIONSHIP</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT

<a href="#">RELATIONSHIP</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">RELATIONSHIP</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">RELATIONSHIP</a>	CONNECTS_TO	CONNECTS_TO Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">RELATIONSHIP</a>	IS_DESCRIBED_BY	IS_DESCRIBED_BY Properties: SEQUENCE_NUMBER OPTIONALITY MINIMUM_PER_SUBJECT MAXIMUM_PER_SUBJECT OBJECT_TEXT
<a href="#">RELATIONSHIP</a>	RELATION_IS_RELATED_VIA_RELATION	RELATION_IS_RELATED_VIA_REL SEQUENCE_NUMBER (1-999) BIG CARDINAL OPTION DEPENDENT ABSTRACT ROLE OBJECT_TEXT
<a href="#">REPORT</a>	OWNS_VIEW	OWNS_VIEW Properties: SEQUENCE_NUMBER VIEW_USAGE OBJECT_TEXT
<a href="#">REPORT</a>	HAS_BITMAP	HAS_BITMAP Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">REPORT</a>	REFERS_TO	REFERS_TO Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">REPORT</a>	REPORT_CONTAINS_SECTION	REPORT_CONTAINS_SECTION Pr SEQUENCE_NUMBER SECTION_TYPE SEQUENCE_NUMBER_BREAK PAGE_PLACEMENT BREAK_FIELD BREAK_QUALIFIER LEFT_MARGIN PRINT_OPTIONS OBJECT_TEXT
<a href="#">REPORT</a>	REPORT_HAS_FORM	REPORT_HAS_FORM Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">REQUIREMENT</a>	[none]	
<a href="#">RULE</a>	OWNS_VIEW	OWNS_VIEW Properties: SEQUENCE_NUMBER VIEW_USAGE OBJECT_TEXT
<a href="#">RULE</a>	HAS_BITMAP	HAS_BITMAP Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">RULE</a>	REFERS_TO	REFERS_TO Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">RULE</a>	ACCESSES	ACCESSES Properties: SEQUENCE_NUMBER COMP_USAGE OBJECT_TEXT

<a href="#">RULE</a>	RULE_DEPENDS_ON_RULE	RULE_DEPENDS_ON_RULE Properties: SEQUENCE_NUMBER (1-999) RULE_DEPENDS_ON_RULE_DESC INCLUSIVE_FLAG EXOR_SEQUENCE_NUMBER
<a href="#">RULE</a>	RULE_CONVERSES_REPORT	RULE_CONVERSES_REPORT Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">RULE</a>	USES_COMPONENT	USES_COMPONENT Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">RULE</a>	CONVERSES_WINDOW	CONVERSES_WINDOW Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">RULE</a>	USES_RULE	USES_RULE Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">RULE</a>	RULE_TRIGGERS_EVENT	RULE_TRIGGERS_EVENT Properties: SEQUENCE_NUMBER TYPE CONDITION VIEW_MAPPING OBJECT_TEXT
<a href="#">SECTION</a>	OWNS_VIEW	OWNS_VIEW Properties: SEQUENCE_NUMBER VIEW_USAGE OBJECT_TEXT
<a href="#">SECTION</a>	USES_LANGUAGE	USES_LANGUAGE Properties: SEQUENCE_NUMBER VALUE_SHORT VALUE_LONG OBJECT_TEXT
<a href="#">SERVER</a>	SERVER_CONTAINS_RULE	SERVER_CONTAINS_RULE Properties: SEQUENCE_NUMBER ENTRY_TYPE RULE_OBJ_NAME SERVICE_NAME OBJECT_TEXT
<a href="#">SET</a>	CONTAINS_VALUE	CONTAINS_VALUE Properties: SEQUENCE_NUMBER SYMBOL OBJECT_TEXT
<a href="#">SET</a>	CONTAINS_SYMBOL	CONTAINS_SYMBOL Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">SOFTWARE_PACKAGE</a>	REBUILD_PACKAGE_CONTAINS	REBUILD_PACKAGE_CONTAINS Properties: SEQUENCE_NUMBER SCOPE_TYPE SEED_STATUS HAS_REUSE IS_A_ROOT IS_A_LEAF_PROCESS HAS_CHANGED EXCP_PROC APPLICATION_CONFIG OBJECT_TEXT



<a href="#">SOFTWARE_PACKAGE</a>	REBUILD_PACKAGE_CONTAINS	EBUILD_PACKAGE_CONTAINS Proj SEQUENCE_NUMBER SCOPE_TYPE SEED_STATUS HAS_REUSE IS_A_ROOT IS_A_LEAF_PROCESS HAS_CHANGED EXCP_PROC APPLICATION_CONFIG OBJECT_TEXT
<a href="#">SOFTWARE_PACKAGE</a>	REBUILD_PACKAGE_CONTAINS	EBUILD_PACKAGE_CONTAINS Proj SEQUENCE_NUMBER SCOPE_TYPE SEED_STATUS HAS_REUSE IS_A_ROOT IS_A_LEAF_PROCESS HAS_CHANGED EXCP_PROC APPLICATION_CONFIG OBJECT_TEXT
<a href="#">SOFTWARE_PACKAGE</a>	REBUILD_PACKAGE_PREPARED_BY	REBUILD_PACKAGE_PREPARED_I SEQUENCE_NUMBER SCOPE_TYPE SEED_STATUS HAS_REUSE PREPARE_PLATFORM OS_TYPE DBMS_TYPE CONFIG_UNIT ROWSELECTED START_TIME_STAMP STOP_TIME_STAMP RETURN_CODE OBJECT_TEXT
<a href="#">SOFTWARE_PACKAGE</a>	REBUILD_PACKAGE_PREPARED_BY	REBUILD_PACKAGE_PREPARED_I SEQUENCE_NUMBER SCOPE_TYPE SEED_STATUS HAS_REUSE PREPARE_PLATFORM OS_TYPE DBMS_TYPE CONFIG_UNIT ROWSELECTED START_TIME_STAMP STOP_TIME_STAMP RETURN_CODE OBJECT_TEXT
<a href="#">SOFTWARE_PACKAGE</a>	REBUILD_PACKAGE_PREPARED_BY	REBUILD_PACKAGE_PREPARED_I SEQUENCE_NUMBER SCOPE_TYPE SEED_STATUS HAS_REUSE PREPARE_PLATFORM OS_TYPE DBMS_TYPE CONFIG_UNIT ROWSELECTED START_TIME_STAMP STOP_TIME_STAMP RETURN_CODE OBJECT_TEXT

<a href="#">SOFTWARE_PACKAGE</a>	REBUILD_PACKAGE_PREPARED_BY	REBUILD_PACKAGE_PREPARED_I SEQUENCE_NUMBER SCOPE_TYPE SEED_STATUS HAS_REUSE PREPARE_PLATFORM OS_TYPE DBMS_TYPE CONFIG_UNIT ROWSELECTED START_TIME_STAMP STOP_TIME_STAMP RETURN_CODE OBJECT_TEXT
<a href="#">SOFTWARE_PACKAGE</a>	REBUILD_PACKAGE_PREPARED_BY	REBUILD_PACKAGE_PREPARED_I SEQUENCE_NUMBER SCOPE_TYPE SEED_STATUS HAS_REUSE PREPARE_PLATFORM OS_TYPE DBMS_TYPE CONFIG_UNIT ROWSELECTED START_TIME_STAMP STOP_TIME_STAMP RETURN_CODE OBJECT_TEXT
<a href="#">SOFTWARE_PACKAGE</a>	REBUILD_PACKAGE_PREPARED_BY	REBUILD_PACKAGE_PREPARED_I SEQUENCE_NUMBER SCOPE_TYPE SEED_STATUS HAS_REUSE PREPARE_PLATFORM OS_TYPE DBMS_TYPE CONFIG_UNIT ROWSELECTED START_TIME_STAMP STOP_TIME_STAMP RETURN_CODE OBJECT_TEXT
<a href="#">SOFTWARE_PACKAGE</a>	REBUILD_PACKAGE_PREPARED_BY	REBUILD_PACKAGE_PREPARED_I SEQUENCE_NUMBER SCOPE_TYPE SEED_STATUS HAS_REUSE PREPARE_PLATFORM OS_TYPE DBMS_TYPE CONFIG_UNIT ROWSELECTED START_TIME_STAMP STOP_TIME_STAMP RETURN_CODE OBJECT_TEXT
<a href="#">SOFTWARE_PACKAGE</a>	REBUILD_PACKAGE_PREPARED_BY	REBUILD_PACKAGE_PREPARED_I SEQUENCE_NUMBER SCOPE_TYPE SEED_STATUS HAS_REUSE PREPARE_PLATFORM OS_TYPE DBMS_TYPE CONFIG_UNIT ROWSELECTED START_TIME_STAMP STOP_TIME_STAMP RETURN_CODE OBJECT_TEXT
<a href="#">SOFTWARE_PACKAGE</a>	REBUILD_PACKAGE_PREPARED_BY	REBUILD_PACKAGE_PREPARED_I SEQUENCE_NUMBER SCOPE_TYPE SEED_STATUS HAS_REUSE PREPARE_PLATFORM OS_TYPE DBMS_TYPE CONFIG_UNIT ROWSELECTED START_TIME_STAMP STOP_TIME_STAMP RETURN_CODE OBJECT_TEXT

<a href="#">SOFTWARE_PACKAGE</a>	REBUILD_PACKAGE_PREPARED_BY	REBUILD_PACKAGE_PREPARED_ SEQUENCE_NUMBER SCOPE_TYPE SEED_STATUS HAS_REUSE PREPARE_PLATFORM OS_TYPE DBMS_TYPE CONFIG_UNIT ROWSELECTED START_TIME_STAMP STOP_TIME_STAMP RETURN_CODE OBJECT_TEXT
<a href="#">STATE</a>	HAS_STD_DRAWING	HAS_STD_DRAWING Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">SUCCESS_FACTOR</a>	SUCCESS_FACTOR_SUPPORTED_BY	SUCCESS_FACTOR_SUPPORTED_ SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">SUCCESS_FACTOR</a>	SUCCESS_FACTOR_SUPPORTED_BY	SUCCESS_FACTOR_SUPPORTED_ SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">SUCCESS_FACTOR</a>	SUCCESS_FACTOR_AFFECTS	SUCCESS_FACTOR_AFFECTS Pro SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">SYMBOL</a>	USES_LANGUAGE	USES_LANGUAGE Properties; SEQUENCE_NUMBER VALUE_SHORT VALUE_LONG OBJECT_TEXT
<a href="#">SYSTEM</a>	DATA_CONTENT_DEFINED_BY	DATA_CONTENT_DEFINED_BY Prc SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">SYSTEM</a>	SYSTEM_AFFECTS_DATA_STORE	SYSTEM_AFFECTS_DATA_STORE SEQUENCE_NUMBER CREATES READS UPDATES DELETES COMMENTS OBJECT_TEXT
<a href="#">SYSTEM</a>	IS_REPLACED_BY	IS_REPLACED_BY Properties: SEQUENCE_NUMBER CURRENT PLANNED COMMENTS OBJECT_TEXT

<a href="#">SYSTEM</a>	SYSTEM_SUPPORTS	SYSTEM_SUPPORTS Properties: SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">SYSTEM</a>	SYSTEM_SUPPORTS	SYSTEM_SUPPORTS Properties: SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">SYSTEM</a>	SYSTEM_SUPPORTS	SYSTEM_SUPPORTS Properties: SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">SYSTEM</a>	SYSTEM_SUPPORTS	SYSTEM_SUPPORTS Properties: SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">SYSTEM</a>	SYSTEM_SUPPORTS	SYSTEM_SUPPORTS Properties: SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">SYSTEM</a>	SYSTEM_SUPPORTS	SYSTEM_SUPPORTS Properties: SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">SYSTEM</a>	SYSTEM_SUPPORTS	SYSTEM_SUPPORTS Properties: SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">SYSTEM</a>	SYSTEM_SUPPORTS	SYSTEM_SUPPORTS Properties: SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">SYSTEM</a>	SYSTEM_SUPPORTS	SYSTEM_SUPPORTS Properties: SEQUENCE_NUMBER RESPONSIBLE MAJOR_INV MINOR_INV COMMENTS OBJECT_TEXT
<a href="#">SYSTEM</a>	SYSTEM_REPLACES	SYSTEM_REPLACES Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">SYSTEM</a>	SYSTEM_CONTAINS	SYSTEM_CONTAINS Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">TABLE</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">TABLE</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT
<a href="#">TABLE</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE NUMBER (1-999) OBJECT_TEXT

<a href="#">TABLE</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">TABLE</a>	IMPLEMENTED_BY	IMPLEMENTED_BY Properties: SEQUENCE_NUMBER (1-999) OBJECT_TEXT
<a href="#">TABLE</a>	TABLE_IS_BASED_ON_TABLE	TABLE_IS_BASED_ON_TABLE Prop SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">TABLE</a>	TABLE_HAS_COLUMN	TABLE_HAS_COLUMN Properties: SEQUENCE_NUMBER NULL_INDICATOR DISTINCT DISTINCT_TYPE UPDATE_PCT OBJECT_TEXT
<a href="#">TABLE</a>	TABLE_HAS_KEY	TABLE_HAS_KEY Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">TABLE</a>	TABLE_REFERRED_BY_KEY	TABLE_REFERRED_BY_KEY Prope SEQUENCE_NUMBER REFERENTIAL_INTEGRITY OBJECT_TEXT
<a href="#">TRANSITION</a>	TRIGGERS_LOGICAL_PROCESS	TRIGGERS_LOGICAL_PROCESS P SEQUENCE_NUMBER TRIGGERS_LOGICAL_PROCESS_I OBJECT_TEXT
<a href="#">TRANSITION</a>	RESULTS_IN_STATE	RESULTS_IN_STATE Properties: SEQUENCE_NUMBER RESULTS_IN_STATE_DESCRIPTIC OBJECT_TEXT
<a href="#">TRANSITION</a>	IS_PRECONDITIONED_BY_STATE	IS_PRECONDITIONED_BY_STATE SEQUENCE_NUMBER IS_PRECONDITIONED_BY_STATE_ OBJECT_TEXT
<a href="#">VALUE</a>	[none]	
<a href="#">VERSION</a>	[none]	
<a href="#">VIEW</a>	VIEW_INCLUDES	VIEW_INCLUDES Properties: SEQUENCE_NUMBER OCCURS NULL_INDICATOR OBJECT_TEXT
<a href="#">VIEW</a>	VIEW_INCLUDES	VIEW_INCLUDES Properties: SEQUENCE_NUMBER OCCURS NULL_INDICATOR OBJECT_TEXT
<a href="#">WINDOW</a>	OWNS_VIEW	OWNS_VIEW Properties: SEQUENCE_NUMBER VIEW_USAGE OBJECT_TEXT
<a href="#">WINDOW</a>	HAS_BITMAP	HAS_BITMAP Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">WINDOW</a>	REFERS_TO	REFERS_TO Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">WINDOW</a>	WINDOW_HAS_WINDOW_CONTENT	WINDOW_HAS_WINDOW_CONTEN SEQUENCE_NUMBER OBJECT_TEXT

<a href="#">WINDOW</a>	HAS_HELP_TEXT	HAS_HELP_TEXT Properties: SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">WINDOW</a>	WINDOW_CONTENT_HAS_HELP	WINDOW_CONTENT_HAS_HELP P SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">WINDOW</a>	WINDOW_CONTENT_HAS_PANEL	WINDOW_CONTENT_HAS_PANEL SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">WINDOW_CONTENT</a>	WINDOW_CONTENT_HAS_HELP	WINDOW_CONTENT_HAS_HELP P SEQUENCE_NUMBER OBJECT_TEXT
<a href="#">WINDOW_CONTENT</a>	WINDOW_PANEL_HAS_HELP	WINDOW_PANEL_HAS_HELP Prop SEQUENCE_NUMBER OBJECT_TEXT

## Relationship Types and Properties

Relationship types have properties that you can set or query. In [Relationship Types and Properties](#), the properties of the relationship type are described after the relationship. If a property has a domain of values, it follows the property description.

### *Relationship Types and Properties*

<a href="#">ACCESSES</a>	<a href="#">IS RELATED VIA</a>
<a href="#">APPLICATION CONFIGURATION HAS CONFIGURATION UNIT</a>	<a href="#">IS REPLACED BY</a>
<a href="#">BITMAP_HAS_BITMAP_IMPLEMENTATION</a>	<a href="#">IS TYPED BY</a>
<a href="#">BUSINESS OBJECT OWNS</a>	<a href="#">KEY HAS COLUMN</a>
<a href="#">BUSINESS OBJECT REFINES INTO BUSINESS OBJECT</a>	<a href="#">LOGICAL PROCESS AFFECTS</a>
<a href="#">CELL CONTAINS</a>	<a href="#">LOGICAL_PROCESS_IMPLEMENTED_BY</a>
<a href="#">COMPONENT USES COMPONENT</a>	<a href="#">MACHINE CAN ACCESS MACHINE</a>
<a href="#">CONFIGURATION_UNIT_ENCAPSULATES</a>	<a href="#">ORGANIZATION_SUPPORTS_ENTITY</a>
<a href="#">CONNECTS TO</a>	<a href="#">ORGANIZATION_SUPPORTS_FUNCTION</a>
<a href="#">CONTAINS VALUE</a>	<a href="#">ORGANIZATION_SUPPORTS_PROCESS</a>
<a href="#">CONVERSES WINDOW</a>	<a href="#">OWNS VIEW</a>
<a href="#">DATABASE HAS TABLE</a>	<a href="#">PROCESS DEPENDS ON PROCESS</a>
<a href="#">DATATYPE IS COMPOSED OF DATATYPE</a>	<a href="#">PROCESS_IMPACTS_ENTITY</a>
<a href="#">DATATYPE IS CONSTRAINED BY SET</a>	<a href="#">PROCESS IS CARRIED OUT AT LOCATION</a>
<a href="#">DATA STORE IS REPLACED BY ENTITY</a>	<a href="#">PROCESS REFINES INTO PROCESS</a>
<a href="#">DEPENDS ON LOGICAL PROCESS</a>	<a href="#">PROCESS REPLACES SYSTEM</a>
<a href="#">ENTITY IS DISTRIBUTED AT</a>	<a href="#">REFERS TO</a>
<a href="#">ENTITY IS MODIFIED AT LOCATION</a>	<a href="#">REFINES INTO</a>
<a href="#">EVENT CAUSES</a>	<a href="#">RELATION IS RELATED VIA RELATION</a>
<a href="#">EVENT HAS RULE</a>	<a href="#">REPORT CONTAINS SECTION</a>
<a href="#">EVENT INFLUENCES BUSINESS OBJECT</a>	<a href="#">RESULTS IN STATE</a>
<a href="#">EVENT TRIGGERS</a>	<a href="#">RULE CONVERSES REPORT</a>
<a href="#">FILE IS FORWARDED TO FILE</a>	<a href="#">RULE DEPENDS ON RULE</a>

<a href="#">FSDM_SCHEMA_HAS_FSDM_VALUE</a>	<a href="#">RULE_TRIGGERS_EVENT</a>
<a href="#">FSDM_VALUE_CLASSIFIES_FSDM_SCHEMA</a>	<a href="#">SERVER_CONTAINS_RULE</a>
<a href="#">FUNCTION_INTERSECTS_WITH_ENTITY</a>	<a href="#">SERVER_DERIVES_SERVER</a>
<a href="#">FUNCTION_IS_CARRIED_OUT_AT_LOCATION</a>	<a href="#">SYSTEM_AFFECTS_DATA_STORE</a>
<a href="#">HAS_IDENTIFIER</a>	<a href="#">TABLE_HAS_COLUMN</a>
<a href="#">HAS_STATE</a>	<a href="#">TABLE_HAS_KEY</a>
<a href="#">IDENTIFIER_IS_COMPOSED_OF</a>	<a href="#">TABLE_IS_BASED_ON_TABLE</a>
<a href="#">IMPLEMENTED_BY</a>	<a href="#">TABLE_REFERRED_BY_KEY</a>
<a href="#">INITIATES_EVENT</a>	<a href="#">TRIGGERS_LOGICAL_PROCESS</a>
<a href="#">IS_COMPOSED_OF_ATTRIBUTE</a>	<a href="#">USES_COMPONENT</a>
<a href="#">IS_COMPOSED_OF_EVENT</a>	<a href="#">USES_RULE</a>
<a href="#">IS_COMPOSED_OF_LOGICAL_PROCESS</a>	<a href="#">VIEW_INCLUDES</a>
<a href="#">IS_DEFINED_BY</a>	<a href="#">WINDOW_CONTENT_HAS_HELP</a>
<a href="#">IS_DESCRIBED_BY</a>	<a href="#">WINDOW_CONTENT_HAS_PANEL</a>
<a href="#">IS_KEYED_BY</a>	<a href="#">WINDOW_HAS_BITMAP531</a>
<a href="#">IS_PRECONDITIONED_BY_STATE</a>	<a href="#">WINDOW_HAS_WINDOW_CONTENT</a>

## ACCESSES

### *ACCESSES properties*

SEQUENCE\_NUMBER

## APPLICATION\_CONFIGURATION\_HAS\_CONFIGURATION\_UNIT

### *APPLICATION\_CONFIGURATION\_HAS\_CONFIGURATION\_UNIT properties*

SEQUENCE\_NUMBER

## BITMAP\_HAS\_BITMAP\_IMPLEMENTATION

### *BITMAP\_HAS\_BITMAP\_IMPLEMENTATION properties*

SEQUENCE\_NUMBER

## BUSINESS\_OBJECT\_OWNS

### *BUSINESS\_OBJECT\_OWNS properties*

SEQUENCE\_NUMBER

## BUSINESS\_OBJECT\_REFINES\_INTO\_BUSINESS\_OBJECT

### *BUSINESS\_OBJECT\_REFINES\_INTO\_BUSINESS\_OBJECT properties*

SEQUENCE\_NUMBER

## CELL\_CONTAINS

### *CELL\_CONTAINS properties*

APPLICATION_SERVER_FLAG	COMMUNICATION_GATEWAY_FLAG
MACHINE_IMPLEMENTATION_NAME	MANAGER_FLAG
NETWORK_ID	SEQUENCE_NUMBER

**COMPONENT\_USES\_COMPONENT**

**COMPONENT\_USES\_COMPONENT properties**

SEQUENCE_NUMBER
-----------------

**CONFIGURATION\_UNIT\_ENCAPSULATES**

**CONFIGURATION\_UNIT\_ENCAPSULATES properties**

SEQUENCE_NUMBER
-----------------

**CONNECTS\_TO**

**CONNECTS\_TO properties**

SEQUENCE_NUMBER
-----------------

**CONTAINS\_VALUE**

**CONTAINS\_VALUE properties**

SEQUENCE_NUMBER	SYMBOL
-----------------	--------

**CONVERSES\_WINDOW**

**CONVERSES\_WINDOW properties**

SEQUENCE_NUMBER
-----------------

**DATABASE\_HAS\_TABLE**

**DATABASE\_HAS\_TABLE properties**

ACTIVATION_PERIOD: Duration	AVE_DELETES
AVE_INSERTS	AVE_UPDATES
DONE_FLAG: Boolean	DO_FLAG: Boolean
MAXIMUM_ROWS	MINIMUM_ROWS
SEQUENCE_NUMBER	

**DATATYPE\_IS\_COMPOSED\_OF\_DATATYPE**

**DATATYPE\_IS\_COMPOSED\_OF\_DATATYPE properties**

OCCURS	SEQUENCE_NUMBER
--------	-----------------

**DATATYPE\_IS\_CONSTRAINED\_BY\_SET**

**DATATYPE\_IS\_CONSTRAINED\_BY\_SET properties**

SEQUENCE_NUMBER
-----------------



**DATA\_STORE\_IS\_REPLACED\_BY\_ENTITY*****DATA\_STORE\_IS\_REPLACED\_BY\_ENTITY properties***

COMMENTS	SEQUENCE_NUMBER
----------	-----------------

**DEPENDS\_ON\_LOGICAL\_PROCESS*****DEPENDS\_ON\_LOGICAL\_PROCESS properties***

DEPENDS_ON_LOGICAL_PROCESS_DESCRIPTION	SEQUENCE_NUMBER
--	-----------------

**ENTITY\_IS\_DISTRIBUTED\_AT*****ENTITY\_IS\_DISTRIBUTED\_AT properties***

COMMENTS	MASTER: Boolean
PARTITIONED: Boolean	REORGANIZED: Boolean
REPLICATED: Boolean	SEQUENCE_NUMBER
SUBSET: Boolean	TELEPROC: Boolean
VARIANT: Boolean	

**ENTITY\_IS\_MODIFIED\_AT\_LOCATION*****ENTITY\_IS\_MODIFIED\_AT\_LOCATION properties***

COMMENTS	CREATE: Boolean
DELETE: Boolean	READ: Boolean
UPDATE: Boolean	

**EVENT\_CAUSES*****EVENT\_CAUSES properties***

EVENT_CAUSES_DESCRIPTION	SEQUENCE_NUMBER
--------------------------	-----------------

**EVENT\_HAS\_RULE*****EVENT\_HAS\_RULE properties***

ACTION_HOST	ACTION_TYPE: ActionType
SEQUENCE_NUMBER	VIEW_MAPPING

**EVENT\_INFLUENCES\_BUSINESS\_OBJECT*****EVENT\_INFLUENCES\_BUSINESS\_OBJECT properties***

EVENT_INFLUENCES_BUSINESS_OBJECT_DESCRIPTION	SEQUENCE_NUMBER
--	-----------------

**EVENT\_TRIGGERS*****EVENT\_TRIGGERS properties***

EVENT_TRIGGERS_DESCRIPTION	EXOR_SEQUENCE_NUMBER
INCLUSIVE_FLAG: InclusiveFlag	SEQUENCE_NUMBER

**FILE\_IS\_FORWARDED\_TO\_FILE**

*FILE\_IS\_FORWARDED\_TO\_FILE properties*

DELETE_RULE	INSERT_RULE
SEQUENCE_NUMBER	UPDATE_RULE

**FSDM\_SCHEMA\_HAS\_FSDM\_VALUE**

*FSDM\_SCHEMA\_HAS\_FSDM\_VALUE properties*

SEQUENCE_NUMBER
-----------------

**FSDM\_VALUE\_CLASSIFIES\_FSDM\_SCHEMA**

*FSDM\_VALUE\_CLASSIFIES\_FSDM\_SCHEMA properties*

SEQUENCE_NUMBER
-----------------

**FUNCTION\_INTERSECTS\_WITH\_ENTITY**

*FUNCTION\_INTERSECTS\_WITH\_ENTITY properties*

COMMENTS	CREATE: Boolean
DELETE: Boolean	READ: Boolean
SEQUENCE_NUMBER	UPDATE: Boolean

**FUNCTION\_IS\_CARRIED\_OUT\_AT\_LOCATION**

*FUNCTION\_IS\_CARRIED\_OUT\_AT\_LOCATION properties*

COMMENTS	MAJOR_INV: Boolean
MINOR_INV: Boolean	SEQUENCE_NUMBER

**HAS\_IDENTIFIER**

*HAS\_IDENTIFIER properties*

SEQUENCE_NUMBER
-----------------

**HAS\_STATE**

*HAS\_STATE properties*

SEQUENCE_NUMBER
-----------------

**IDENTIFIER\_IS\_COMPOSED\_OF**

*IDENTIFIER\_IS\_COMPOSED\_OF properties*

SEQUENCE_NUMBER
-----------------

**IMPLEMENTED\_BY**

*IMPLEMENTED\_BY properties*

SEQUENCE_NUMBER
-----------------

**INITIATES\_EVENT**

***INITIATES\_EVENT properties***

INITIATES_EVENT_DESCRIPTION	SEQUENCE_NUMBER
-----------------------------	-----------------

**IS\_COMPOSED\_OF\_ATTRIBUTE**

***IS\_COMPOSED\_OF\_ATTRIBUTE properties***

OCCURS	SEQUENCE_NUMBER
--------	-----------------

**IS\_COMPOSED\_OF\_EVENT**

***IS\_COMPOSED\_OF\_EVENT properties***

SEQUENCE_NUMBER
-----------------

**IS\_COMPOSED\_OF\_LOGICAL\_PROCESS**

***IS\_COMPOSED\_OF\_LOGICAL\_PROCESS properties***

SEQUENCE_NUMBER
-----------------

**IS\_DEFINED\_BY**

***IS\_DEFINED\_BY properties***

SEQUENCE_NUMBER
-----------------

**IS\_DESCRIBED\_BY**

***IS\_DESCRIBED\_BY properties***

SEQUENCE_NUMBER	OPTIONALITY: Boolean
-----------------	----------------------

**IS\_KEYED\_BY**

***IS\_KEYED\_BY properties***

SEQUENCE_NUMBER	TYPE: KeyType
-----------------	---------------

**IS\_PRECONDITIONED\_BY\_STATE**

***IS\_PRECONDITIONED\_BY\_STATE properties***

IS_PRECONDITIONED_BY_STATE_DESCRIPTION	SEQUENCE_NUMBER	
--	-----------------	--

**IS\_RELATED\_VIA**

***IS\_RELATED\_VIA properties***

ABSTRACT: Boolean	CARDINALITY: Cardinal
CONTROLLING: Boolean	DEPENDENT: Boolean
MAXIMUM_CARDINALITY	MINIMUM_CARDINALITY
OPTIONALITY: Boolean	ROLE
SEQUENCE_NUMBER	

**IS\_REPLACED\_BY*****IS\_REPLACED\_BY properties***

COMMENTS	CURRENT: Boolean
PLANNED: Boolean	SEQUENCE_NUMBER

**IS\_TYPED\_BY*****IS\_TYPED\_BY properties***

SEQUENCE_NUMBER
-----------------

**KEY\_HAS\_COLUMN*****KEY\_HAS\_COLUMN properties***

SEQUENCE_NUMBER
-----------------

**LOGICAL\_PROCESS\_AFFECTS*****LOGICAL\_PROCESS\_AFFECTS properties***

COMMENTS	CREATES: Boolean
DELETES: Boolean	READS: Boolean
SEQUENCE_NUMBER	UPDATES: Boolean

**LOGICAL\_PROCESS\_IMPLEMENTED\_BY*****LOGICAL\_PROCESS\_IMPLEMENTED\_BY properties***

SEQUENCE_NUMBER
-----------------

**MACHINE\_CAN\_ACCESS\_MACHINE*****MACHINE\_CAN\_ACCESS\_MACHINE properties***

SEQUENCE_NUMBER
-----------------

**ORGANIZATION\_SUPPORTS\_ENTITY*****ORGANIZATION\_SUPPORTS\_ENTITY properties***

COMMENTS	CREATE: Boolean
DELETE: Boolean	READ: Boolean
SEQUENCE_NUMBER	UPDATE: Boolean

**ORGANIZATION\_SUPPORTS\_FUNCTION*****ORGANIZATION\_SUPPORTS\_FUNCTION properties***

COMMENTS	MAJOR_INVOLVEMENT: Boolean
MINOR_INVOLVEMENT: Boolean	RESPONSIBLE
SEQUENCE_NUMBER	

**ORGANIZATION\_SUPPORTS\_PROCESS**

***ORGANIZATION\_SUPPORTS\_PROCESS properties***

COMMENTS	MAJOR_INVOLVEMENT: Boolean
MINOR_INVOLVEMENT: Boolean	RESPONSIBLE: Boolean
SEQUENCE_NUMBER	

**OWNS\_VIEW**

***OWNS\_VIEW properties***

SEQUENCE_NUMBER	VIEW_USAGE: ViewUsage
-----------------	-----------------------

**PROCESS\_DEPENDS\_ON\_PROCESS**

***PROCESS\_DEPENDS\_ON\_PROCESS properties***

PROCESS_DEPENDS_ON_PROCESS_DESCRIPTION	EXOR_SEQUENCE_NUMBER
INCLUSIVE_FLAG: InclusiveFlag	SEQUENCE_NUMBER

**PROCESS\_IMPACTS\_ENTITY**

***PROCESS\_IMPACTS\_ENTITY properties***

COMMENTS	CREATE: Boolean
DELETE: Boolean	READ: Boolean
SEQUENCE_NUMBER	UPDATE: Boolean

**PROCESS\_IS\_CARRIED\_OUT\_AT\_LOCATION**

***PROCESS\_IS\_CARRIED\_OUT\_AT\_LOCATION properties ||***

COMMENTS	MAJOR_INVOLVEMENT: Boolean
MINOR_INVOLVEMENT: Boolean	SEQUENCE_NUMBER

**PROCESS\_REFINES\_INTO\_PROCESS**

<b><i>PROCESS_REFINES_INTO_PROCESS properties</i></b>	
CONDITIONAL_FLAG	SEQUENCE_NUMBER

**PROCESS\_REPLACES\_SYSTEM**

***PROCESS\_REPLACES\_SYSTEM properties***

SEQUENCE_NUMBER
-----------------

**REFERS\_TO**

***REFERS\_TO properties***

SEQUENCE_NUMBER
-----------------

**REFINES\_INTO**

**REFINES\_INTRO properties**

SEQUENCE_NUMBER
-----------------

**RELATION\_IS\_RELATED\_VIA\_RELATION**

**RELATION\_IS\_RELATED\_VIA\_RELATION properties**

SEQUENCE_NUMBER
-----------------

**REPORT\_CONTAINS\_SECTION**

**REPORT\_CONTAINS\_SECTION properties**

BREAK_FIELD	BREAK_QUALIFIER
LEFT_MARGIN	PAGE_PLACEMENT: PagePlace
PRINT_OPTIONS: PrintOpt	SECTION_TYPE: SectionType
SEQUENCE_NUMBER	SEQUENCE_NUMBER_BREAK

**RESULTS\_IN\_STATE**

**RESULTS\_IN\_STATE properties**

RESULTS_IN_STATE_DESCRIPTION	SEQUENCE_NUMBER
------------------------------	-----------------

**RULE\_CONVERSES\_REPORT**

**RULE\_CONVERSES\_REPORT properties**

SEQUENCE_NUMBER
-----------------

**RULE\_DEPENDS\_ON\_RULE**

**RULE\_DEPENDS\_ON\_RULE properties**

RULE_DEPENDS_ON_RULE_DESCRIPTION	EXOR_SEQUENCE_NUMBER
INCLUSIVE_FLAG	SEQUENCE_NUMBER

**RULE\_TRIGGERS\_EVENT**

**RULE\_TRIGGERS\_EVENT properties**

CONDITION	SEQUENCE_NUMBER
TYPE: RuleTriggerType	VIEW_MAPPING

**SERVER\_CONTAINS\_RULE**

**SERVER\_CONTAINS\_RULE properties**

ENTRY_TYPE: EntryType	RULE_OBJ_NAME
SEQUENCE_NUMBER	SERVICE_NAME

**SERVER\_DERIVES\_SERVER**

**SERVER\_DERIVES\_SERVER properties**

---

SEQUENCE\_NUMBER

**SYSTEM\_AFFECTS\_DATA\_STORE**

***SYSTEM\_AFFECTS\_DATA\_STORE properties***

COMMENTS	CREATE: Boolean
DELETE: Boolean	READ: Boolean
SEQUENCE_NUMBER	UPDATE: Boolean

**TABLE\_HAS\_COLUMN**

***TABLE\_HAS\_COLUMN properties***

DISTINCT	DISTINCT_TYPE: DistinctType
NULL_INDICATOR: Null	SEQUENCE_NUMBER

**TABLE\_HAS\_KEY**

***TABLE\_HAS\_KEY properties***

SEQUENCE\_NUMBER

**TABLE\_IS\_BASED\_ON\_TABLE**

***TABLE\_IS\_BASED\_ON\_TABLE properties***

SEQUENCE\_NUMBER

**TABLE\_REFERRED\_BY\_KEY**

***TABLE\_REFERRED\_BY\_KEY properties***

REFERENTIAL\_INTEGRITY: Boolean SEQUENCE\_NUMBER

**TRIGGERS\_LOGICAL\_PROCESS**

***TRIGGERS\_LOGICAL\_PROCESS properties***

TRIGGERS\_LOGICAL\_PROCESS\_DESCRIPTION SEQUENCE\_NUMBER

**USES\_COMPONENT**

***USES\_COMPONENT properties***

SEQUENCE\_NUMBER

**USES\_RULE**

***USES\_RULE properties***

SEQUENCE\_NUMBER

**VIEW\_INCLUDES**

***VIEW\_INCLUDES properties***

---

NULL_INDICATOR: NullInd	OCCURS
SEQUENCE_NUMBER	

#### WINDOW\_CONTENT\_HAS\_HELP

##### *WINDOW\_CONTENT\_HAS\_HELP properties*

SEQUENCE\_NUMBER

#### WINDOW\_CONTENT\_HAS\_PANEL

##### *WINDOW\_CONTENT\_HAS\_PANEL properties*

SEQUENCE\_NUMBER

#### WINDOW\_HAS\_BITMAP531

##### *WINDOW\_HAS\_BITMAP properties*

SEQUENCE\_NUMBER

#### WINDOW\_HAS\_WINDOW\_CONTENT

##### *WINDOW\_HAS\_WINDOW\_CONTENT properties*

SEQUENCE\_NUMBER

## TurboCycler Window Controls

Controls are the visual elements in a window with which an end user can interact. Windows Controls provides instructions for coding window blocks and supplies the reference data to complete the window control definitions in [TurboCycler Template Language](#).

- [Window Statement Control Types](#)
- [Window Control Properties](#)
- [Property Types and Property Values](#)
- [Window Properties Matrix](#)

To design a window using the TurboCycler template language, create controls and arrange them in a window. In addition to controls, a window can have a menu bar with pull-down menus.

## Creating Controls

When you create a control with a MAKE statement, you also set its properties. Controls generally require positioning properties such as LEFT, BOTTOM, WIDTH, and HEIGHT. Any property not explicitly set defaults to settings similar to those in Window Painter.

Specify colors by either of two methods. The first is to set the RGB color value explicitly; the other is to set a property to a color listed under colortype. The first example uses the three RGB colors:

```
MAKE STATIC_TEXT
HAVING [ HPSID = "ID_TEXT_1",
        BGCOLOR_RED = 255,
        BGCOLOR_GREEN = 0,
        BGCOLOR_BLUE = 0 ]
ENDMAKE
```

The second example uses an explicit color type definition that produces the same result.



```

MAKE STATIC_TEXT

HAVING [ HPSID = "ID_TEXT_2",
BGCOLOR = "RED" ]
ENDMAKE

```

However, you cannot use both methods to set the same color property. For example, you can set the background color of a control with the three RGB properties BGCOLOR\_RED, BGCOLOR\_GREEN, and BGCOLOR\_BLUE, or you can set it with the single colortype property BGCOLOR. To establish a relationship between a SPREADSHEET and its CELLS, complete the following steps:

1. Create your spreadsheet with a MAKE statement.
2. Associate each of its cells with the spreadsheet through the CONTAIN statement, as shown.

```

MAKE SPREADSHEET Spreadsheet
HAVING [BOTTOM = 40,
LEFT = 20,
HEIGHT = 155,
WIDTH = 200,
HPSID = "ID_SPREADSHEET" ]
LINKED TO MyDataView
ENDMAKE

CONTAIN CELL IN Spreadsheet
HAVING [WIDTH = CellWidth,
HEIGHT = 22,
HPSID = (QUERY NAME OF MyDataFields(I)),
CONTROL_TEXT = (QUERY SCREEN_LITERAL OF
MyDataFields(I)),
IMMEDIATE_RETURN = "TRUE" ]
LINKED TO MyDataFields(I)
ENDCONTAIN

```

You need the same relationship for LB\_COLUMN (which must be in a LISTBOX) and CHART\_X\_AXIS and CHART\_Y\_AXIS (which must be in a CHART\_WINDOW).

3. When TABSTOP is TRUE, the tabbing order for controls is the same order as they are created in the window block.

The following sections list control types and their properties of the window statement. Also see [Window Properties](#) for a summary of all the control types and properties.

## Property Types and Property Values

This section details the contents of each of the window block properties in TurboCycler, as listed in [Window Block Property Types](#). The supported values for each property type are listed in the tables under each subheading. Refer to [Window Control Properties](#) for cross-reference of support for properties and controls.

### Window Block Property Types

<a href="#">boolean</a>	<a href="#">fonttype</a>
<a href="#">charformattype</a>	<a href="#">funckeytype</a>
<a href="#">charttype</a>	<a href="#">justificationtype</a>
<a href="#">colortype</a>	<a href="#">modifiertype</a>
<a href="#">drawlinestyle</a>	<a href="#">selectmodetype</a>
<a href="#">fieldtype</a>	

### boolean

### boolean Property Types

FALSE	false
NO	no
TRUE	true
YES	yes
0	1

**charformattype**

**charformattype Property Types**

ALLFIRSTUPPER	FIRSTUPPER
LOWER	UPPER

**charttype**

**charttype Property Types**

AREACHART2D	AREACHART3D
BARCHART2D	BARCHART3D
BARLINECHART2D	BARLINECHART3D
CANDLE2D	COLUMNCHART2D
COLUMNCHART3D	HILOCLOSE2D
LINECHART2D	LINECHART3D
PERBARCHART3D	PIECHART2D
PIECHART3D	POINTANDFIG2D
SMOOTHLINECHART2D	SCATTERCHART2D
STACKEDBARCHART2D	STACKEDBARCHART3D

**colortype**

**colortype Property Types**

BLACK	BLUE
BROWN	CYAN
DARK_BLUE	DARK_CYAN
DARK_GRAY	DARK_GREEN
DARK_MAGENTA	DARK_RED
DARK_YELLOW	GRAY
GREEN	MAGENTA
PINK	RED
WHITE	YELLOW

**drawlinestyle**

**drawlinestyle Properties**

HLINES	NOLINES
--------	---------

VHLINES	VLINES
---------	--------

**fieldtype**

**fieldtype Properties**

EDIT	PROTECTED
CONTROL_TEXT	

**fonttype**

**fonttype Properties**

MODERN8	MODERN10
MODERN12	ROMAN8
ROMAN10	ROMAN12
ROMAN14	ROMAN18
ROMAN24	SWISS8
SWISS10	SWISS12
SWISS14	SWISS18
SWISS24	SYSTEMFONT8

**funckeytype**

**funckeytype Properties**

F1	F2
F3	F4
F5	F6
F7	F8
F9	F10
F11	F12

**justificationtype**

**justificationtype Properties**

CENTER	LEFT
RIGHT	

**modifiertype**

**Modifiertype Properties**

ALT	CTRL	
SHIFT		

**selectmodetype**

**Selectmodetype properties**

---

EXTENDED	MULTIPLE
SINGLE	

## Window Control Properties

[Window Control Properties](#) lists all the properties and the property types for window block controls. The linked subtopics describe each property in alphabetical order. Refer to [Property Types and Property Values](#) for more details. For a cross-referenced chart of all window controls and properties, refer to the [Window Properties Matrix](#).

The window control properties in TurboCycler are listed in the following table:

### Window Control Properties

<a href="#">A</a>	<a href="#">I-J-K-L</a>
<a href="#">B</a>	<a href="#">M-N-O-P</a>
<a href="#">C-D-E</a>	<a href="#">R</a>
<a href="#">E</a>	<a href="#">S through Z</a>
<a href="#">G-H</a>	

### A

#### **ASCII\_KEY string**

Specifies an ASCII character to be used as the shortcut key for a control. If you specify ASCII\_KEY, set MODIFIER to CTRL. (Refer to [MODIFIER modifyertype](#) and [FUNC\\_KEY funckeytype](#).)

#### **AUTO\_CALL boolean**

If AUTO\_CALL is TRUE for a LISTBOX or SPREADSHEET, control returns to the invoking rule when the end user tries to scroll beyond the first or last occurrence defined in the view data structure associated with the list box. The rule can then retrieve more data. Set AUTO\_CALL only if you do not use the component SET\_VIRTUAL\_LISTBOX\_SIZE described in the AppBuilder documentation for your system.

#### **AUTO\_SELECT boolean**

If AUTO\_SELECT is TRUE, the end user can put the cursor on an item in a SPREADSHEET to select it. Any selected item is deselected. If AUTO\_SELECT is FALSE, the end user can move the cursor without changing the selection. AUTO\_SELECT does not apply when SELECT\_MODE is set to MULTIPLE. (See [SELECT\\_MODE selectmodetype](#).)

### B

#### **BDCOLOR colortype**

Sets the border color of the control to one of the colors listed for colortype.

#### **BDCOLOR\_RED integer**

Sets the border color RGB red value. Assign an integer value between 0 and 255.

#### **BDCOLOR\_GREEN integer**

Sets the border color RGB green value. Assign an integer value between 0 and 255.

#### **BDCOLOR\_BLUE integer**

Sets the border color RGB blue value. Assign an integer value between 0 and 255.

#### **BGCOLOR colortype**

Sets the background color of the control to one of the colors listed for colortype.

#### **BGCOLOR\_RED integer**

Sets the background color RGB red value. Assign an integer value between 0 and 255.

***BGCOLOR\_GREEN integer***

Sets the background color RGB green value. Assign an integer value between 0 and 255.

***BGCOLOR\_BLUE integer***

Sets the background color RGB blue value. Assign an integer value between 0 and 255.

***BLUE integer***

Sets the control color RGB blue value. Assign an integer value between 0 and 255.

***BOTTOM integer***

Sets the vertical position of the bottom edge of the control.

**C-D-E**

***CELL\_HEIGHT integer***

Sets the vertical size of a CELL in a SPREADSHEET.

***CHARFORMAT charformattype***

Sets the case of text in the control.

***CHART\_TYPE charttype***

Sets the type of chart displayed in a CHART\_WINDOW control to one of the types listed under charttype.

***CHECK\_MANDATORY\_FIELDS boolean***

If TRUE, all controls that have their MANDATORY property set to TRUE must contain valid entries when the end user selects the PUSH\_BUTTON. If a control does not contain a valid entry, a message box is displayed and the first control containing an invalid entry receives the input focus. (See [MANDATORY boolean](#).)

***COLOR colortype***

Sets the control color to one of the colors listed for colortype.

***CONTROL\_TEXT string***

Specifies the label that appears on CELL, CHART\_WINDOW, CHECKBOX, GROUPBOX, PUSH\_BUTTON, RADIO\_BUTTON, or STATIC\_TEXT controls. To define a mnemonic key, precede the character with an ampersand (&) or a tilde (~).

***DRAW\_LINES drawlinestype***

Determines whether a SPREADSHEET is displayed with horizontal and vertical lines, horizontal lines only, vertical lines only, or no lines at all.

**F**

***FGCOLOR colortype***

Sets the foreground color of the control to one of the colors listed for colortype.

***FGCOLOR\_RED integer***

Sets the foreground color RGB red value. Assign an integer value between 0 and 255.

***FGCOLOR\_GREEN integer***

Sets the foreground color RGB green value. Assign an integer value between 0 and 255.

***FGCOLOR\_BLUE integer***

Sets the foreground color RGB blue value. Assign an integer value between 0 and 255.

**FIELD\_TEXT string**

If the FIELD\_TYPE property of a CELL were CONTROL\_TEXT, specifies the text to be displayed. If the FIELD\_TYPE is not CONTROL\_TEXT, do not set this property.

**FIELD\_TYPE fieldtype**

Specifies the type of field displayed in a CELL: text, edit, or protected edit.

**FILE string**

For a BITMAP, identifies the bit map file associated with the control; for a DROPDOWN\_COMBOBOX, specifies the Reference Table Name of the field entity linked to the DROPDOWN\_COMBOBOX with a .REF extension. The Reference Table Name attribute should be the system ID of the online validation set associated with the field entity in the hierarchy.

**FONT fonttype**

Sets the font of labels and input text. Use the default font where possible to make the windows device independent. Not all controls support all fonts. Selecting a font can cause the HEIGHT property to be ignored for EDIT\_FIELD, PROTECTED\_EDIT\_FIELD, RADIO\_BUTTON, and CHECKBOX controls.

**FOOTING string**

Sets the footing text displayed in a CHART\_WINDOW.

**FOOTING\_COLOR colortype**

Sets the color of the FOOTING text displayed in a CHART\_WINDOW.

**FOOTING\_COLOR\_RED integer**

Sets the RGB color red value of the FOOTING text displayed in a CHART\_WINDOW. Assign an integer value between 0 and 255.

**FOOTING\_COLOR\_GREEN integer**

Sets the RGB color green value of the FOOTING text displayed in a CHART\_WINDOW. Assign an integer value between 0 and 255.

**FOOTING\_COLOR\_BLUE integer**

Sets the RGB color blue value of the FOOTING text displayed in a CHART\_WINDOW. Assign an integer value between 0 and 255.

**FUNC\_KEY funckeytype**

Sets the function key to be used as a shortcut key for the control. (See [ASCII\\_KEY string](#).)

**G-H****GREEN integer**

Sets the control color RGB green value. Assign an integer value in the range of 0 to 255.

**GROUPSTART boolean**

If TRUE, specifies that the control starts a group. Do not set this property for other controls in the group. Create all controls in the group in sequence so that the tab order will be correct.

**HEADER\_BOTTOM integer**

Specifies the vertical position of the column headings in a SPREADSHEET control.

**HEADER\_HEIGHT integer**

Specifies the height of the column headings in a SPREADSHEET control.

**HEADER\_HPSID string**

Specifies the HPSID for static text in a multicolumn list box header.

***HEADING string***

Sets heading text displayed in a CHART\_WINDOW.

***HEADING\_COLOR colortype***

Sets the color of the HEADING text displayed in a CHART\_WINDOW.

***HEADING\_COLOR\_RED integer***

Sets the RGB color red value of the HEADING text displayed in a CHART\_WINDOW. Assign an integer value between 0 and 255.

***HEADING\_COLOR\_GREEN integer***

Sets the RGB color green value of the HEADING text displayed in a CHART\_WINDOW. Assign an integer value between 0 and 255.

***HEADING\_COLOR\_BLUE integer***

Sets the RGB color blue value of the HEADING text displayed in a CHART\_WINDOW. Assign an integer value between 0 and 255.

***HEIGHT integer***

Sets the vertical size of the control.

***HPSID string***

Contains the text that uniquely identifies the control to the system. Every HPSID must be unique for the generated application to work correctly at run time.

**I-J-K-L**

***IMMEDIATE\_RETURN boolean***

If TRUE, this control returns control to the invoking rule when the end user navigates away from the control. For a read-only control, double-click the control to return program control to the invoking rule.

***JUSTIFICATION justificationtype***

This property formats control text as right justified, left justified, or centered. This property applies to all the field types: text, numeric, date, and time.

***LEFT integer***

Sets the horizontal position of the control.

***LEFTLABEL string***

Sets the text displayed on the left side of a CHART\_WINDOW.

***LEFTLABEL\_COLOR colortype***

Sets the color of the LEFTLABEL text displayed in a CHART\_WINDOW.

***LEFTLABEL\_COLOR\_RED integer***

Sets the RGB color red value of the LEFTLABEL text displayed in a CHART\_WINDOW. Assign an integer value between 0 and 255.

***LEFTLABEL\_COLOR\_GREEN integer***

Sets the RGB color green value of the LEFTLABEL text displayed in a CHART\_WINDOW. Assign an integer value between 0 and 255.

***LEFTLABEL\_COLOR\_BLUE integer***

Sets the RGB color blue value of the LEFTLABEL text displayed in a CHART\_WINDOW. Assign an integer value between 0 and 255.

***LEGEND string***

Specifies the legend label for a CHART\_Y\_AXIS control in a CHART\_WINDOW.

## M-N-O-P

### **MANDATORY** *boolean*

If TRUE, requires that the control contain valid input when the end user selects a PUSH\_BUTTON having CHECK\_MANDATORY\_FIELDS set to TRUE. (See [CHECK\\_MANDATORY\\_FIELDS](#) *boolean*.)

### **MAXIMIZE** *boolean*

If TRUE, a maximize arrow appears on the window of a CHART\_WINDOW.

### **MAXIMUM** *decimal*

Sets the maximum value allowed for numeric fields.

### **MINIMIZE** *boolean*

If TRUE, a minimize arrow appears on the window of a CHART\_WINDOW.

### **MINIMUM** *decimal*

Sets the minimum value allowed for numeric fields.

### **MODIFIER** *modifiertype*

Specifies that the Ctrl, Alt, or Shift keys must be pressed with a FUNC\_KEY or ASCII\_KEY to activate a shortcut key for a control. You can use only one modifier. For a ASCII\_KEY, set the MODIFIER to CTRL. (See [ASCII\\_KEY](#) *string* and [FUNC\\_KEY](#) *funckeytype*.)

### **NUMBERING\_RECORD** *boolean*

When TRUE, the rows of a SPREADSHEET control are numbered on the left side of the control.

### **PICTURE** *string*

Sets the display format of fields. The string consists of literal text and format specifiers that control the appearance of date, time, and numeric fields, as shown in [Date Specifiers](#) through [Numeric and Replacement Characters](#). For the date and time fields, the picture can combine literal text with the specifiers, as shown in [Standard Date Formats](#) and [Standard Time Formats](#).

#### **Date Specifiers**

Specifier	Description
%c	Century
%D	Day of the month, with suffix
%d	Day of the month
%j	Julian day
%M	Month of the year, text
%m	Month of the year
%W	Day of the week, text
%Y	Year
%y	Years since 1900

#### **Standard Date Formats**

Date	Example
%m/%d/%y	1/5/95
%0m/%0d/%0y	01/05/95



%m-%d	1-5
%m-%y	1-95
%0c%0y%0m%0d	19950105
%W, %M %D, %Y	Thursday, January 5th, 1995
Today is %W.	Today is Thursday
%d/%m/%y	1/5/95
%0d/%0m/%y	01/05/95
%d-%m	5-1
%j	5
%c	19

For time fields, the picture can contain literal text alone and in combination with the specifiers, as shown in [Time Format Specifiers](#) and [Standard Time Formats](#).

**Time Format Specifiers**

Specifier	Description
%H	Hour, text (24-hour clock)
%h	Hour (12-hour clock)
%M	Minutes, text
%m	Minutes
%s	Seconds
%t	Hour (24-hour clock)
%x	AM/PM indicator

**Standard Time Formats**

Time display picture	Example
%0h:%0m:%0s %x	01:22:03 PM
%0h:%0m:%0s	13:22:03
%h-%m-%s %x	1-22-3 PM
%t-%m-%s	13-22-3
%h %x	1 PM
%H %M	Thirteen Twenty_Two
Time is %H	Time is Thirteen
The time is %h-%m-%s %x.	The time is 1-22-3 PM.

For numeric fields, valid replacement characters are: 9 . , \$ Z \* S + - cr db, as shown in [Numeric and Replacement Characters](#). The country setting determines the replacement of the characters: . , \$ S.

**Numeric and Replacement Characters**

Character	Description
9	Numeric character
.	Decimal separator

,	Thousands separator	
\$	Currency symbol	
Z	Suppress leading zeroes	
*	Pad left with asterisks	
S	Sign character	
+	Sign character, printed only if positive	
-	Sign character, printed only if negative	
cr	Credit symbol, printed only if negative	
db	Debit symbol, printed only if negative	

## R

### ***READ\_ONLY*** *boolean*

If TRUE, the end user cannot change the text in the control.

### ***RED*** *integer*

Sets the control color RGB red value. Assign an integer value between 0 and 255.

### ***RESIZE*** *boolean*

If TRUE, end users can grab the border of a CHART\_WINDOW to resize the chart.

### ***RIGHTLABEL*** *string*

Sets the text displayed on the right side of a CHART\_WINDOW.

### ***RIGHTLABEL\_COLOR*** *colortype*

Sets the color of the RIGHTLABEL text displayed in a CHART\_WINDOW.

### ***RIGHTLABEL\_COLOR\_RED*** *integer*

Sets the RGB color red value of the RIGHTLABEL text displayed in a CHART\_WINDOW. Assign an integer value between 0 and 255.

### ***RIGHTLABEL\_COLOR\_GREEN*** *integer*

Sets the RGB color green value of the RIGHTLABEL text displayed in a CHART\_WINDOW. Assign an integer value between 0 and 255.

### ***RIGHTLABEL\_COLOR\_BLUE*** *integer*

Sets the RGB color blue value of the RIGHTLABEL text displayed in a CHART\_WINDOW. Assign an integer value between 0 and 255.

### ***ROW\_SELECT*** *boolean*

If TRUE, the end user can select an entire row in a SPREADSHEET by clicking a cell in a row. Otherwise, only the clicked cell is selected.

## S through Z

### ***SCROLL\_LOCK*** *boolean*

If TRUE, the end user can tab to the next row in a SPREADSHEET when reaching the end of the current row; also, the end user must press Ctrl+Tab to shift the input focus to the next control in the window. If FALSE, the end user's tabbing at the end of a row shifts the input focus to the next control in the window.

### ***SELECT\_MODE*** *selectmodetype*

Specifies in a SPREADSHEET whether the end user can select only one item (SINGLE), zero or more items (MULTIPLE), or one or more items (EXTENDED).

**TABSTOP boolean**

If TRUE, the end user can tab to the control. The order that you specify objects in the window block determines the tabbing order of window controls in the completed window.

**VISIBLE boolean**

If TRUE, the control is visible to the end user.

**WIDTH integer**

Sets the horizontal width of the control. For a CELL, this is the width of the column in the SPREADSHEET.

**WORDWRAP boolean**

If TRUE, automatically wraps text displayed in a MULTILINE\_EDIT to a new line, based on the size of the control. If FALSE, the end user must scroll horizontally to see hidden text.

## Window Properties Matrix

The following table provides a comprehensive matrix indicating supported properties and controls for windows:

**Window Properties**

Properties	Controls:					
	STATIC_TEXT	SPREADSHEET	RECTANGLE	RADIO_BUTTON	PUSH_BUTTON	PROTECTED_E
AUTO_CALL		x				
AUTO_SELECT		x				
BDCOLOR	x	x		x	x	x
BDCOLOR_RED	x	x		x	x	x
BDCOLOR_GREEN	x	x		x	x	x
BDCOLOR_BLUE	x	x		x	x	x
BGCOLOR	x	x		x	x	x
BGCOLOR_RED	x	x		x	x	x
BGCOLOR_GREEN	x	x		x	x	x
BGCOLOR_BLUE	x	x		x	x	x
BLUE			x			
BOTTOM	x	x	x	x	x	x
CELL_HEIGHT		x				
CHARFORMAT						x
CHART_TYPE						
CHECK_MANDATORY_FIELDS					x	
COLOR			x			
DRAW_LINES		x				
FGCOLOR	x	x		x	x	x
FGCOLOR_RED	x	x		x	x	x
FGCOLOR_GREEN	x	x		x	x	x
FGCOLOR_BLUE	x	x		x	x	x

FIELD_TEXT						
FIELD_TYPE						
FILE						
FONT	x	x		x	x	x
FOOTING						
FOOTING_COLOR						
FOOTING_COLOR_RED						
FOOTING_COLOR_GREEN						
FOOTING_COLOR_BLUE						
FUNC_KEY					x	
GREEN			x			
GROUPSTART		x		x	x	x
HEADER_BOTTOM						
HEADER_HEIGHT		x				
HEADER_HPSID		x				
HEADING						
HEADING_COLOR						
HEADING_COLOR_RED						
HEADING_COLOR_GREEN						
HEADING_COLOR_BLUE						
HEIGHT	x	x	x	x	x	x
HPSID	x	x	x	x	x	x
IMMEDIATE_RETURN				x		x
JUSTIFICATION						x
ASCII_KEY					x	
LEFT	x	x	x	x	x	x
LEFTLABEL						
LEFTLABEL_COLOR						
LEFTLABEL_COLOR_RED						
LEFTLABEL_COLOR_GREEN						
LEFTLABEL_COLOR_BLUE						
LEGEND						
MANDATORY						
MAXIMIZE						
MAXIMUM						x
MINIMIZE						
MINIMUM						x
MODIFIER					x	
NUMBERING_RECORD		x				

PICTURE						x
READ_ONLY						
RED			x			
RESIZE						
RIGHTLABEL						
RIGHTLABEL_COLOR						
RIGHTLABEL_COLOR_RED						
RIGHTLABEL_COLOR_GREEN						
RIGHTLABEL_COLOR_BLUE						
ROW_SELECT		x				
SCROLL_LOCK		x				
SELECT_MODE		x				
TABSTOP		x		x	x	x
CONTROL_TEXT	x			x	x	
VISIBLE	x	x		x	x	x
WIDTH	x	x	x	x	x	x
WORDWRAP						

## Window Statement Control Types

Window controls are the visual elements in a window with which the end user interacts. The window control types and their properties are described in this topic. Refer to [Window Properties Matrix](#) for a cross-referenced list of all window controls and properties. The window controls in TurboCycler are listed in the following table:

### *TurboCycler Window Controls*

<a href="#">BITMAP</a>	<a href="#">GROUPBOX</a>
<a href="#">CELL</a>	<a href="#">HOTSPOT</a>
<a href="#">CHART_WINDOW</a>	<a href="#">LB_COLUMN</a>
<a href="#">CHART_X_AXIS</a>	<a href="#">LISTBOX</a>
<a href="#">CHART_Y_AXIS</a>	<a href="#">MULTILINE_EDIT</a>
<a href="#">CHECKBOX</a>	<a href="#">PROTECTED_EDIT_FIELD</a>
<a href="#">DROPDOWN_COMBOBOX</a>	<a href="#">PUSH_BUTTON</a>
<a href="#">DROPDOWN_LISTBOX</a>	<a href="#">RADIO_BUTTON</a>
<a href="#">EDIT_FIELD</a>	<a href="#">RECTANGLE</a>
<a href="#">ELLIPSE</a>	<a href="#">SPREADSHEET</a>
<a href="#">FILE_EDITOR</a>	<a href="#">STATIC_TEXT</a>

### **BITMAP**

A special type of static control that displays a bitmap from a file. It cannot be linked to a repository object. Its properties are as follows:

### **BITMAP Control Properties**

BOTTOM	FILE
--------	------

HEIGHT	HPSID
LEFT	VISIBLE
WIDTH	

### **CELL**

A CELL represents a column in a SPREADSHEET. It is not a visual control. The linked-to field must be in a multiply-occurring logical view. A CELL can be created only by a CONTAIN statement. Its properties are as follows:

#### **CELL Properties**

CHARFORMAT	FIELD_TEXT
FIELD_TYPE	HEADER_BOTTOM
HEIGHT	HPSID
IMMEDIATE_RETURN	JUSTIFICATION
MANDATORY	MAXIMUM
MINIMUM	PICTURE
CONTROL_TEXT	WIDTH

### **CHART\_WINDOW**

Displays data graphically in one of thirteen different formats. End users cannot tab to a chart. The CHART\_WINDOW must be linked to a multiply-occurring view. Data to be displayed is specified by creating CHART\_X\_AXIS and CHART\_Y\_AXIS controls using the CONTAIN statement. A window can contain no more than ten CHART\_WINDOW controls. Its properties are as follows:

#### **CHART\_WINDOW Properties**

BOTTOM	CHART_TYPE
FOOTING	FOOTING_COLOR
FOOTING_COLOR_RED	FOOTING_COLOR_GREEN
FOOTING_COLOR_BLUE	HEADING
HEADING_COLOR	HEADING_COLOR_RED
HEADING_COLOR_GREEN	HEADING_COLOR_BLUE
HEIGHT	HPSID
LEFT	LEFTLABEL
LEFTLABEL_COLOR	LEFTLABEL_COLOR_RED
LEFTLABEL_COLOR_BLUE	LEFTLABEL_COLOR_GREEN
MAXIMIZE	MINIMIZE
RESIZE	RIGHTLABEL
RIGHTLABEL_COLOR	RIGHTLABEL_COLOR_RED
RIGHTLABEL_COLOR_GREEN	RIGHTLABEL_COLOR_BLUE
CONTROL_TEXT	WIDTH

### **CHART\_X\_AXIS**

CHART\_X\_AXIS represents an x-axis variable in a CHART\_WINDOW. It is not really a visual control. This control must be created with the CONTAIN statement. The linked-to field must be in a multiply-occurring view and can be of any type. Only one CHART\_X\_AXIS control is created per CHART\_WINDOW. Its property is as follows:

#### **CHART\_X\_AXIS Properties**

HPSID
-------

### **CHART\_Y\_AXIS**

CHART\_Y\_AXIS represents a y-axis variable in a CHART\_WINDOW. It is not really a visual control. This control must be created with the CONTAIN statement. The linked-to field must be in a multiply-occurring view and must be of numeric type. A CHART\_WINDOW can contain up to eight CHART\_Y\_AXIS controls. Its properties are as follows:

#### **CHART\_y\_AXIS Properties**

HPSID	LEGEND
-------	--------

### **CHECKBOX**

A toggle switch you can use when its setting is independent of any other control. The linked-to field type is character, and its length must be one. You can specify a mnemonic key with the CONTROL\_TEXT property. Its properties are:

#### **CONTROL\_TEXT Properties**

BDCOLOR	BDCOLOR_RED
BDCOLOR_GREEN	BDCOLOR_BLUE
BGCOLOR	BGCOLOR_RED
BGCOLOR_GREEN	BGCOLOR_BLUE
BOTTOM	FGCOLOR
FGCOLOR_RED	FGCOLOR_GREEN
FGCOLOR_BLUE	FONT
GROUPSTART	HEIGHT
HPSID	IMMEDIATE_RETURN
LEFT	TABSTOP
CONTROL_TEXT	VISIBLE
WIDTH	

### **DROPDOWN\_COMBOBOX**

DROPDOWN\_COMBOBOX combines an EDIT\_FIELD and a LISTBOX. The user can type a value in the control or select a value from a scrollable list. The linked-to field must have its Reference Table Name attribute set to the System ID of the online validation set associated with the field entity in the hierarchy. Its properties are as follows:

#### **DROPDOWN\_COMBOBOX Properties**

BDCOLOR	BDCOLOR_RED
BDCOLOR_GREEN	BDCOLOR_BLUE
BGCOLOR	BGCOLOR_RED
BGCOLOR_GREEN	BGCOLOR_BLUE
BOTTOM	FGCOLOR
FGCOLOR_RED	FGCOLOR_GREEN
FGCOLOR_BLUE	FILE
FONT	GROUPSTART
HEIGHT	HPSID
IMMEDIATE_RETURN	LEFT
MANDATORY	TABSTOP

VISIBLE	WIDTH
---------	-------

***DROPDOWN\_LISTBOX***

DROPDOWN\_LISTBOX behaves like the DROPDOWN\_COMBOBOX except that the user can type values only from the set referenced by the Reference Table Name attribute of the linked-to field. Its properties are as follows:

***DROPDOWN\_LISTBOX Properties***

BDCOLOR	BDCOLOR_RED
BDCOLOR_GREEN	BDCOLOR_BLUE
BGCOLOR	BGCOLOR_RED
BGCOLOR_GREEN	BGCOLOR_BLUE
BOTTOM	FGCOLOR
FGCOLOR_RED	FGCOLOR_GREEN
FGCOLOR_BLUE	FILE
FONT	GROUPSTART
HEIGHT	HPSID
IMMEDIATE_RETURN	LEFT
MANDATORY	TABSTOP
VISIBLE	WIDTH

***EDIT\_FIELD***

Defines an entry field in which the end user can type and edit data associated with the linked-to field, which can be a single occurrence field of any type except TIMESTAMP. Its properties are as follows:

***EDIT\_FIELD Properties***

BDCOLOR	BDCOLOR_RED
BDCOLOR_GREEN	BDCOLOR_BLUE
BGCOLOR	BGCOLOR_RED
BGCOLOR_GREEN	BGCOLOR_BLUE
BOTTOM	CHARFORMAT
FGCOLOR	FGCOLOR_RED
FGCOLOR_GREEN	FGCOLOR_BLUE
FONT	GROUPSTART
HEIGHT	HPSID
IMMEDIATE_RETURN	JUSTIFICATION
LEFT	MANDATORY
MAXIMUM	MINIMUM
PICTURE	TABSTOP
VISIBLE	WIDTH

***ELLIPSE***

A special type of static control that displays a filled ellipse in the window. It cannot be linked to a repository object. Its properties are as follows:

***ELLIPSE Properties***



BLUE	BOTTOM
COLOR	GREEN
HEIGHT	HPSID
LEFT	RED
WIDTH	

### ***FILE\_EDITOR***

End users can view the contents of a text file, but they cannot edit it. The linked-to field must be of type character or variable character and must contain the complete path name of the file to display at run time. Its properties are:

#### ***FILE\_EDITOR Properties***

BDCOLOR	BDCOLOR_RED
BDCOLOR_GREEN	BDCOLOR_BLUE
BGCOLOR	BGCOLOR_RED
BGCOLOR_GREEN	BGCOLOR_BLUE
BOTTOM	FGCOLOR
FGCOLOR_RED	FGCOLOR_GREEN
FGCOLOR_BLUE	FONT
GROUPSTART	HEIGHT
HPSID	LEFT
TABSTOP	VISIBLE
WIDTH	WORDWRAP

### ***GROUPBOX***

Associates related controls, usually RADIO\_BUTTONs. Within the group, the end user can use the arrow keys rather than Tab to navigate among controls. A GROUPBOX cannot be linked to a repository object. Its properties are:

#### ***GROUPBOX Properties***

BOTTOM	CONTROL_TEXT
FGCOLOR	FGCOLOR_RED
FGCOLOR_GREEN	FGCOLOR_BLUE
FONT	HEIGHT
HPSID	LEFT
VISIBLE	WIDTH

### ***HOTSPOT***

Behaves like a PUSH\_BUTTON but is not displayed at run time. Usually, a HOTSPOT is positioned behind a BITMAP to create what appears to end users as a bitmap push button. Like a PUSH\_BUTTON, a HOTSPOT cannot be linked to a repository object. Its properties are:

#### ***HOTSPOT Properties***

BOTTOM	HEIGHT
HPSID	LEFT
WIDTH	

### ***LB\_COLUMN***

LB\_COLUMN represents a column in a LISTBOX. It is not really a visual control. The linked-to field must be in a multiply-occurring logical view. You must create one LB\_COLUMN for each LISTBOX using the CONTAIN statement. Its properties are as follows:

#### ***LB\_COLUMN Properties***

CHARFORMAT	JUSTIFICATION
MAXIMUM	MINIMUM
PICTURE	

### ***LISTBOX***

Displays a list of choices in a vertically scrolling window. A LISTBOX is linked to a multiply-occurring view. The field entity is specified by creating a LB\_COLUMN in a CONTAIN statement. Its properties are as follows:

#### ***LISTBOX Properties***

AUTO_CALL	BDCOLOR
BDCOLOR_RED	BDCOLOR_GREEN
BDCOLOR_BLUE	BGCOLOR
BGCOLOR_RED	BGCOLOR_GREEN
BGCOLOR_BLUE	BOTTOM
FGCOLOR	FGCOLOR_RED
FGCOLOR_GREEN	FGCOLOR_BLUE
FONT	GROUPSTART
HEIGHT	HPSID
IMMEDIATE_RETURN	LEFT
SELECT_MODE	TABSTOP
VISIBLE	WIDTH

### ***MULTILINE\_EDIT***

Defines an entry field in which the user can enter text on multiple lines. The linked-to field must be of the type character or variable character. Its properties are as follows:

#### ***MULTILINE\_EDIT Properties***

BDCOLOR	BDCOLOR_RED
BDCOLOR_GREEN	BDCOLOR_BLUE
BGCOLOR	BGCOLOR_RED
BGCOLOR_GREEN	BGCOLOR_BLUE
BOTTOM	FGCOLOR
FGCOLOR_RED	FGCOLOR_GREEN
FGCOLOR_BLUE	FONT
GROUPSTART	HEIGHT
HPSID	IMMEDIATE_RETURN
LEFT	MANDATORY
READ_ONLY	TABSTOP

VISIBLE	WIDTH
WORDWRAP	

**PROTECTED\_EDIT\_FIELD**

Behaves like an EDIT\_FIELD except that the end user cannot edit the displayed data. Its properties are as follows:

**PROTECTED\_EDIT\_FIELD Properties**

BDCOLOR	BDCOLOR_RED
BDCOLOR_GREEN	BDCOLOR_BLUE
BGCOLOR	BGCOLOR_RED
BGCOLOR_GREEN	BGCOLOR_BLUE
BOTTOM	CHARFORMAT
FGCOLOR	FGCOLOR_RED
FGCOLOR_GREEN	FGCOLOR_BLUE
FONT	GROUPSTART
HEIGHT	HPSID
IMMEDIATE_RETURN	JUSTIFICATION
LEFT	MAXIMUM
MINIMUM	PICTURE
TABSTOP	VISIBLE
WIDTH	

**PUSH\_BUTTON**

PUSH\_BUTTON usually completes an end user's interaction with the window. When the end user selects a PUSH\_BUTTON, its HPSID is returned to the invoking rule. You cannot link a PUSH\_BUTTON to a repository object.

There are two methods used to associate push buttons with shortcut keys:

- Specify the key as an ASCII character using the ASCII\_KEY property.
- Use the function keys listed under the funckeytype property domain with the FUNC\_KEY property.

You cannot specify the same button with both methods. However, you can use one method for one button and the other method for a different button.

You can also set the MODIFIER property to require that the Ctrl, Alt, or Shift key be pressed simultaneously with the ASCII\_KEY or FUNC\_KEY property. For Window Painter compatibility, you must set MODIFIER to CTRL if the shortcut is an ASCII\_KEY. You can use only one modifier, so, for example, attempting to combine the Ctrl+Tab keys is an error. You can specify a PUSH\_BUTTON mnemonic by the CONTROL\_TEXT property.

The PUSH\_BUTTON properties are listed in [PUSH\\_BUTTON Properties](#).

**PUSH\_BUTTON Properties**

ASCII_KEY	FGCOLOR_GREEN
BDCOLOR	FGCOLOR_RED
BDCOLOR_BLUE	FONT
BDCOLOR_GREEN	FUNC_KEY
BDCOLOR_RED	GROUPSTART
BGCOLOR	HEIGHT
BGCOLOR_BLUE	HPSID
BGCOLOR_GREEN	LEFT

BGCOLOR_RED	MODIFIER
BOTTOM	TABSTOP
CONTROL_TEXT	VISIBLE
FGCOLOR	WIDTH
FGCOLOR_BLUE	

***RADIO\_BUTTON***

A toggle switch that indicates one of a mutually exclusive set of choices. The linked-to field has type character and a length sufficient to hold the longest HPSID of the RADIO\_BUTTONS in the group. You can specify a mnemonic key with the CONTROL\_TEXT property. Its properties are listed in the following table:

***RADIO\_BUTTON Properties***

BDCOLOR	FGCOLOR_GREEN
BDCOLOR_BLUE	FGCOLOR_RED
BDCOLOR_GREEN	FONT
BDCOLOR_RED	GROUPSTART
BGCOLOR	HEIGHT
BGCOLOR_BLUE	HPSID
BGCOLOR_GREEN	IMMEDIATE_RETURN
BGCOLOR_RED	LEFT
BOTTOM	TABSTOP
CONTROL_TEXT	VISIBLE
FGCOLOR	WIDTH
FGCOLOR_BLUE	

***RECTANGLE***

A special type of static control that displays a filled rectangle on the window. It cannot be linked to a repository object. Its properties are listed in the following table:

***RECTANGLE Properties***

BLUE	HPSID
BOTTOM	LEFT
COLOR	RED
GREEN	WIDTH
HEIGHT	

***SPREADSHEET***

Contains columns that represent linked fields and rows that represent each field occurrence. The end user can edit cells thus formed. The SPREADSHEET must be linked to a multiply-occurring logical view. Specify the fields displayed in each column by creating a CELL control using the CONTAIN statement for each field. This control only supports the following fonts: SYSTEMFONT8, SWISS8, ROMAN8, MODERN8, MODERN10, and MODERN12. Its properties are listed in the following table:

***SPREADSHEET Properties***

AUTO_CALL	FGCOLOR_RED
AUTO_SELECT	FONT

BDCOLOR	GROUPSTART
BDCOLOR_BLUE	HEADER_HEIGHT
BDCOLOR_GREEN	HEADER_HPSID
BDCOLOR_RED	HEIGHT
BGCOLOR	HPSID
BGCOLOR_BLUE	LEFT
BGCOLOR_GREEN	NUMBERING_RECORD
BGCOLOR_RED	ROW_SELECT
BOTTOM	SCROLL_LOCK
CELL_HEIGHT	SELECT_MODE
DRAW_LINES	TABSTOP
FGCOLOR	VISIBLE
FGCOLOR_BLUE	WIDTH
FGCOLOR_GREEN	

### **STATIC\_TEXT**

Displays labels or instructions. A STATIC\_TEXT cannot be linked to a repository object. Its properties are listed in the following table:

#### **STATIC\_TEXT Properties**

BDCOLOR	FGCOLOR
BDCOLOR_BLUE	FGCOLOR_BLUE
BDCOLOR_GREEN	FGCOLOR_GREEN
BDCOLOR_RED	FGCOLOR_RED
BGCOLOR	FONT
BGCOLOR_BLUE	HEIGHT
BGCOLOR_GREEN	HPSID
BGCOLOR_RED	LEFT
BOTTOM	VISIBLE
CONTROL_TEXT	WIDTH

## **Build Scripts**

In addition to the TurboScripter objects described [TurboScripter Object's Model Reference](#), the following objects are also available for the build scripts.

- [BuildFramework Object](#)
- [ApplicationConfiguration Object](#)
- [PartitionConfiguration Object](#)
- [BuildConfiguration Object](#)

## **The new AppBuilder Object Oriented Development**

The build framework uses build scripts to build an entity. It uses the AppBuilder's TurboScripting host as a scripting engine. The Turbo scripting engine supports two scripting languages, Javascript (JScript) and Visual Basic (VB) script. Build Framework also supports XSLT scripts. You can define your own build passes for a type in the build configuration file and then write the steps required to execute the pass in Javascript.

**[Example: ODF Generation for an Entity](#)**

The following is a sample build script to generate ODF for an Entity at build time.

```
//file odfgenerator.js
//Repository Object APPCFG
var objAppCfg = InputObject(0);

//Repository object PARTITION
var objPartition = InputObject(1);

//Current Object
var curObject = InputObject(2);

//The main configuration object
var BuildAppConfig = BuildFramework.GetApplicationConfiguration();

//The interface to partition odf properties
var BuildPartConfig = BuildAppConfig.GetPartitionConfiguration(objPartition.GetProperty("ShortName"));

var genLanguage = BuildPartConfig.GetProperty("GeneratedLanguage");
var LogSubDir = BuildConfig.GetProperty(genLanguage, "LogDir"); //log
var odfSubDir = BuildConfig.GetProperty(genLanguage, "OdfDir");
var workDir = BuildPartConfig.GetWorkingDirectory();

var odfPath = workDir\+ "
\\
" \+odfSubDir;

var objShortName = curObject.GetProperty("ShortName");

var logFile = workDir + "
\\
" +LogSubDir+
"
\\
OdfGen.out";
Trace.OutFile = logFile;

Trace.Log("OdfGen.js :%s\n", "Entering");

var msg = curObject.GetProperty("Name");
msg \+= " ";
msg \+= objShortName;
msg \+= ".odf.xml";
Trace.Log("OdfGen.js :generating ODF - %s\n", msg);

//extract Odf for current Object
curObject.ExtractODF(odfPath);

var result = 0; //???
var fso = new ActiveXObject("Scripting.FileSystemObject");
WriteResult(fso, workDir, LogSubDir, result);

Trace.Log("OdfGen.js :%s\n", "Exiting");

function WriteResult(fso, workingDir, logDir, rslt)
\{

var ForWriting = 2;
var outfile2 = workingDir;
outfile2 \+= "
\\
";
outfile2 \+= logDir;
outfile2 \+= "
\\
OdfGen.js.out";
```

```
Trace.Log("Writing Results :%s\n", outfile2);
```

```
var odfResult = fso.OpenTextFile(outfile2, ForWriting, true, \-2);  
odfResult .Write(rsIt);
```

```
odfResult .Close();  
  
\}
```

## BuildFramework Object

This object implements the ITSBuildFramework interface. BuildFramework object represents the OO Development build/preparation framework. Users who write build scripts can access this object from the build scripts. This object's properties are defined below.

### GetWorkingDirectory(String partition\_short\_name)

This method returns the root build directory for a partition, given its shortname.

Example:

```
var work_dir = BuildFramework.GetWorkingDirectory("ZBADGJFT")
```

### GetApplicationConfiguration()

This method returns the application configuration object.

## ApplicationConfiguration Object

This object represents the Build application configuration object. It implements the ITSBuildFramework. For building/preparing an application, you have to configure the application using the Application configuration, which can contain multiple partitions. The script can access this object and the following properties.

### GetPartitionConfiguration(String partition\_short\_name)

This method returns the Partition configuration object.

### IsRemoteRule(String rulename)

Returns true if the given rule is a remote rule in the application.

## PartitionConfiguration Object

This object represents the partition in the OO development build framework. The script can access the following properties of this object.

### GetWorkingDirectory()

This method gets the current build directory.

### GetProperty(String PropName)

This method gets any property of the PARTITION object.

### GetDistributionDirectory()

Gets the distribution directory for the application (distribution directory is the directory where the binary distributables are copied to when building an application).

## BuildConfiguration Object

This object represents the build configuration. The configuration is loaded from the BuildConfig.xml file. The build script can get the properties defined in the build configuration file. The build configuration defines how to build an AppBuilder type. There is one set of configuration options for a build target language such as Java, C#.

### GetProperty(PropName)

This method gets the named property of the current configuration, from the build configuration xml file.

Here is a list of the properties supported for Java:

- RootDir
- DataDir
- ScriptDir



- LogDir
- OutputDir
- CodeGenerator
- CodegenOptions
- MaxCodegenInput
- SourceExtension
- Compiler
- CompilerOptions
- OutputExtension
- Classpath
- MaxCompilerInput
- OdfDir
- BinDir
- MiscDir
- Archiver
- ArchiverOptions
- ArchiverOutputExtension
- WsGen
- WsdOutDir
- WsImport
- WsdFile
- DestDir
- BuildDir
- WsArchive
- WsArchiveExtn