

VisualAge Generator Templates



# Introducing VisualAge Generator Templates

*Version 4.5*



VisualAge Generator Templates



# Introducing VisualAge Generator Templates

*Version 4.5*

**Note**

Before using this document, read the general information under “Notices” on page v.

**First Edition (August 2000)**

This edition applies to the following licensed programs:

- VisualAge Generator Templates Version 4.5

Order publications by phone or fax. IBM Software Manufacturing Solutions takes publication orders between 8:30 a.m. and 7:00 p.m. eastern standard time (EST). The phone number is (800) 879-2755. The fax number is (800) 284-4721.

You can also order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for comments appears at the back of this publication. If the form has been removed, address your comments to:

IBM SWS - Paris Laboratory  
1, place Jean-Baptiste Clément  
93881 Noisy-le-Grand CEDEX  
France.

You can fax comments to +33 1 49 31 57 91 (provisional).

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1997, 2000. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>Notices.</b> . . . . .	<b>v</b>	<b>Chapter 7. Application Definition</b> . . . . .	<b>23</b>
<b>Trademarks.</b> . . . . .	<b>vii</b>	Data Element: Description Completion . . . . .	23
<b>About this document.</b> . . . . .	<b>ix</b>	Value Style: Data Element Presentation Style . . . . .	24
Conventions Used in this Book. . . . .	ix	Business Object: Application Data View. . . . .	26
Other VAGTemplates documentation . . . . .	ix	Interface Unit: End User Interface. . . . .	27
		Help . . . . .	28
<b>Part 1. General Information</b> . . . . .	<b>1</b>	<b>Chapter 8. VAGTemplates Generation.</b> . . . . .	<b>31</b>
<b>Chapter 1. What 's New in VAGTemplates</b>		Generation Overview . . . . .	31
<b>4.5</b> . . . . .	<b>3</b>	VAGTemplates on Smalltalk 3.x Generators . . . . .	31
Functional Evolutions . . . . .	3	VAGTemplates on Smalltalk Generators. . . . .	31
Information Model . . . . .	3	VAGTemplates on Java Generators . . . . .	32
Instance Menu . . . . .	3	Generated Beans/Parts . . . . .	32
Workspace . . . . .	3	<b>Chapter 9. Generator Customization</b> . . . . .	<b>35</b>
Interface Unit Definition . . . . .	3	Entity Extension . . . . .	35
Business Object Definition. . . . .	3	Generator Customization. . . . .	36
Computing Join Condition . . . . .	3		
Relational Table Definition . . . . .	3	<b>Part 2. Getting Started</b> . . . . .	<b>39</b>
Duplicate Instance . . . . .	3	<b>Chapter 10. Introduction</b> . . . . .	<b>41</b>
Enhanced Functions in Generated GUI Java . . . . .	4	<b>Chapter 11. Starting VAGTemplates and</b>	
Traceability. . . . .	4	<b>Defining a New Workspace</b> . . . . .	<b>45</b>
<b>Chapter 2. VisualAge Generator Templates</b>		<b>Chapter 12. Importing a Relational</b>	
<b>Overview</b> . . . . .	<b>5</b>	<b>Database.</b> . . . . .	<b>49</b>
VAGTemplates Workbench . . . . .	6	<b>Chapter 13. Exploring the Pre-filled</b>	
VAGTemplates Information Model . . . . .	6	<b>Instances</b> . . . . .	<b>51</b>
VAGTemplates Generators . . . . .	7	<b>Chapter 14. Defining the Data Element</b>	
VAGTemplates Additional Features. . . . .	7	<b>Labels and Input Checks</b> . . . . .	<b>55</b>
<b>Chapter 3. Development Steps</b> . . . . .	<b>11</b>	<b>Chapter 15. Creating a Business Object.</b> . . . . .	<b>61</b>
<b>Chapter 4. Presentation of the Workbench</b>	<b>13</b>	<b>Chapter 16. Defining a Business Object . . . . .</b>	<b>63</b>
<b>Chapter 5. The Workspace</b> . . . . .	<b>15</b>	Definition . . . . .	63
Independence of Functional Description and		Generation Parameters . . . . .	65
Generation Parameters . . . . .	15	<b>Chapter 17. Creating an Interface Unit</b>	
Setting Parameters . . . . .	16	<b>Using the Sample Business Object as a</b>	
<b>Chapter 6. Relational Database Import</b> . . . . .	<b>19</b>	<b>Detail</b> . . . . .	<b>67</b>
Relational Table: Imported Data Storage. . . . .	19		
Data Element: Imported Elementary			
Information . . . . .	20		

<b>Chapter 18. Defining an Interface Unit . . .</b>	<b>69</b>	<b>Chapter 21. Testing your Application . . .</b>	<b>75</b>
Definition . . . . .	69	Testing the GUI Client Application . . . . .	75
Generation Parameters . . . . .	70	Testing the TUI Application . . . . .	77
<b>Chapter 19. TUI Only: Creating and</b>		<b>Chapter 22. Exploring the Generated Parts</b>	<b>83</b>
<b>Defining a Menu . . . . .</b>	<b>71</b>	Exploring the Parts Generated by the GUI	
Creating the Menu Interface Unit . . . . .	71	Generators . . . . .	83
Defining the Menu Interface Unit . . . . .	71	Exploring the Parts Generated by the TUI	
Definition . . . . .	71	Generators . . . . .	85
Generation Parameters . . . . .	72		
 		<hr/>	
<b>Chapter 20. Generating your Application</b>	<b>73</b>	<b>Part 3. Appendixes . . . . .</b>	<b>89</b>
Generating your GUI Client Application . . . . .	73		
Generating your TUI Application . . . . .	74	<b>Parameterizing DB2 Database Import. . . . .</b>	<b>91</b>

---

## Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk NY 10504-1785, U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact the SWS General Legal Counsel, IBM Corporation, Department TL3 Building 062, P. O. Box 12195, Research Triangle Park, NC 27709-2195. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

IBM has made reasonable efforts to ensure the accuracy of the information contained in this publication. If a softcopy of this publication is provided to you with the product, you should consider the information contained in the softcopy version the most recent and most accurate. However, this publication is presented "as is" and IBM makes no warranties of any kind with respect to the contents hereof, the products listed herein, or the completeness or accuracy of this publication.

IBM may change this publication, the product described herein, or both.





---

## Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

DB2

IBM

OS/2

VisualAge

The following terms are trademarks of other companies:

Microsoft, Windows, VisualBasic, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.



---

## About this document

If you are new to VisualAge Generator Templates, we suggest that you read this book first for an overview of its features and capabilities.

This book is divided into two parts.

The first part provides general information on VisualAge Generator Templates, as a Rapid Architected Application Development feature of VisualAge Generator. It includes both an overview of its objectives and functions and the description of the main steps involved in the application development cycle.

The second part gets you started with VAGTemplates, guiding you through the different steps required for building a simple application, from the relational database import to the application generation and test.

---

## Conventions Used in this Book

In this document, VisualAge Generator Templates is abbreviated to VAGTemplates.

References to other parts of the documentation or other VAGTemplates manuals are written in *italics*.

The symbols used in this documentation are:

**Tip**                      Tip or helpful hint

**Warning**                Proceed with caution (risky or irreversible action, etc.)

---

## Other VAGTemplates documentation

*VisualAge Generator Templates Standard Functions — User's Guide*

The first part of this document presents the VAGTemplates Workbench and its tools.

The second part describes the standard use of the Workbench:

- It shows you how to build operational applications using the standard functions of VAGTemplates.
- It gives you illustrated documentation on the functions and behavior of the generated applications.

- It explains how to generate an application and provides a description of the generated components and their role within the generated application.

*VisualAge Generator Templates Customizer — User's Guide*

This document includes both the standard and advanced functions of VAGTemplates. It describes the generation technology at work in VAGTemplates and guides you through generator customization and creation.

*VisualAge Generator Templates Customizer on Java — Reference Guide*

This document which is intended for the user of the generator customization functions in the VisualAge for Java environment describes the API of the following VAGTemplates components:

- The Information Model API
- The Generation Framework API
- The generator API
- The APIs used when building VisualAge Generator parts and VisualAge for Java components

*VisualAge Generator Templates Customizer on Smalltalk — Reference Guide*

This document which is intended for the user of the generator customization functions in the VisualAge Smalltalk Enterprise environment describes the API of the following VAGTemplates components:

- The Information Model API
- The Generation Framework API
- The generator API
- The APIs used when building VisualAge Generator and VisualAge Smalltalk Enterprise parts

---

## Part 1. General Information



---

# Chapter 1. What 's New in VAGTemplates 4.5

---

## Functional Evolutions

### Information Model

A new attribute, **Long Target Name**, enlarges the possibilities to name Smalltalk or Java generated class prefix, as it bears more characters.

### Instance Menu

New functionalities are available through the **Instance Menu**:

- **References**: displays a window with the instances that are referencing the selected instances.
- **Set Generation Parameters to Default**
- **Create Business Object from RT**
- **Create Interface Unit from BO**

### Workspace

It is possible to automatically get **SQL high value** and **SQL low value** from the Database, for alphanumeric extractions.

### Interface Unit Definition

The windows used to add Interface Units and Business Objects in the Interface Unit and in the Business Object pages enable the creation of the object to be added.

### Business Object Definition

The windows used to add volatile fields (in the Table and Fields page) enable, the creation of DataElement instances.

### Computing Join Condition

A 'Compute' button in the SQL Join Condition page enables the automatic computing of a Join Condition (default initialization), from 'n' mapped tables.

### Relational Table Definition

In the Columns page, it is possible to update the 'Updatable' attribute (for Views, only). In the Tables page, primary and secondary keys can be created, columns can be added or removed from key; links can be either added or removed. The Target Table can be opened.

### Duplicate Instance

The package is now displayed in the Duplicate Instance window.

---

## Enhanced Functions in Generated GUI Java

- The edit and window list functions are now integrated to the GUI Java (cut/copy/paste, windows), and a prompt on close is displayed;
- Bidirectional specifications are available for Java, when arabic or latin characters are needed;
- On-line help is available from the Help Menu;
- The management of exceptions is enhanced.

### Traceability

Traceability provides information about the source model component that is responsible for the specification of a generated component. Traceability also determines the rules that manage the collision between the component to store and the existing component in the target tool.

It is now possible to get to know traceability information through comments in the generated Java code.



---

## Chapter 2. VisualAge Generator Templates Overview

VAGTemplates is a Rapid Architected Application Development add-on feature of VisualAge Generator.

VAGTemplates automates the development of business applications by generating application components from existing database definitions.

These definitions populate the VAGTemplates **Information Model** which allows you to formalize high-level specifications via its dedicated entities.

The integration of VisualAge Generator and VAGTemplates enables you to easily and rapidly produce a complete operational client / server application:

- VAGTemplates enables you to build GUI client components in either a VisualAge for Java or VisualAge Smalltalk Enterprise environment,
- VAGTemplates enables you to build both server and TUI client components.

**Note:** Instances of the Information Model entities and the VisualAge Generator application components are stored in the VisualAge Library. This central library includes advanced team development facilities such as version control, multi-user support and release management.

The generated components may be easily enhanced by adding specific business logic using VisualAge Generator. Additions to the generated components are preserved, thus providing the ability to maintain the functional specifications at the VAGTemplates Information Model level.

The Information Model is open; its entities may be extended to meet specific business needs.

The VAGTemplates generation engine is an open object framework that enables you to customize the VAGTemplates standard generators or to create new ones.

VAGTemplates is available in two options:

- VAGTemplates *Standard Functions*, the base product, is dedicated to developing applications with the **standard** VAGTemplates generators.

**Note:** This option is packaged with VisualAge Generator at no additional charge.

- VAGTemplates *Customizer*, is used to customize generation.

**Notes:**

1. This option requires a specific license and installation which is separately orderable and priced.
2. The Standard Functions are included in this option.

---

## VAGTemplates Workbench

The VAGTemplates Workbench is a multi-windowed graphical interface that enables you to import a relational database, model your application, edit your application specifications, and generate applications. In VisualAge Generator, you can then modify the automatically built graphical user interfaces in the Composition Editor to enhance the generated application.

The VAGTemplates Workbench is an integrated feature of your VisualAge Generator environment. You open the VAGTemplates Workbench from a menu in your VisualAge Generator workstation.

---

## VAGTemplates Information Model

The VAGTemplates Information Model provides you with a number of entities for defining business applications that are target-system independent. Instances of the Information Model entities are reusable and enable you to define applications in a consistent, non-redundant, and systematic manner.

Using the import function, VAGTemplates enables you to automatically integrate an existing **relational database** from the catalogs. The import function creates instances of two Information Model entities: the Relational Table and the Data Element. When you create a **Business Object** and an **Interface Unit**, you set data manipulation specifications for your application.

**Note:** VAGTemplates supports DB2 and Oracle 7 database management systems.

You can extend the entities of the Information Model to meet specific business needs.

The VAGTemplates Information Model can be populated from other modeling tools using the open API, either in Java or in Smalltalk environments.

The following diagram presents the basic Information Model entities and the relationships that exist between them.

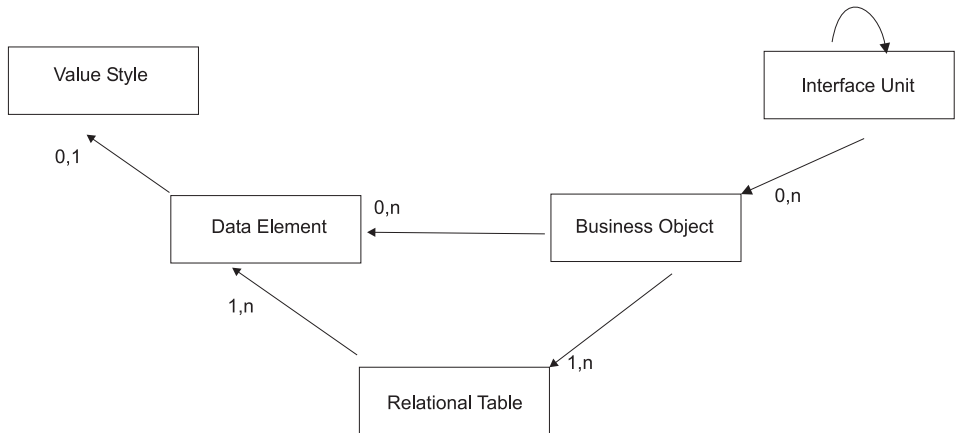


Figure 1. VAGTemplates Information Model: Entities and Relationships

## VAGTemplates Generators

The VAGTemplates generators convert the application specifications into fully operational VisualAge Generator Web, GUI (Graphical User Interface) components, TUI (Text User Interface) components, and server components.

VAGTemplates automatically generates all the components you need to:

- Access and manipulate data in databases (create, read, update, delete),
- Present the data to the end-user,
- Manage navigation among multiple windows,
- Manage multi-user data access concurrency,
- Manage paging and scrolling of data presented in a list,
- Manage error conditions,
- Provide the end-user with an on-line help facility.

The VAGTemplates automatic layout function provides well-designed user interfaces for your applications. You only have to specify a few presentation parameters and let the VAGTemplates' automatic layout function create template layouts for the graphic components or maps being generated.

The generators are open. You can customize the standard generators or create new ones by modifying and reusing their code in the VisualAge workstation. This enables you to incorporate and automatically ensure adherence to site standards in the generated applications.

## VAGTemplates Additional Features

The VAGTemplates reverse engineering function automatically builds a generator from an application that you have developed manually using VisualAge.

VAGTemplates ensures that prototyping does not produce "throw-away" components. Any code generated during prototyping remains part of the application as it develops.

VAGTemplates can be used in a variety of national language environments, including DBCS environments. Corresponding parameters appear at the user interface level, enabling the generation of national language Web, GUI or TUI applications.

VAGTemplates provides you with a migration tool to upgrade specifications defined with VAGTemplates V3.0 or V3.1.

For more information on the version migration, refer to the VisualAge Generator Migration Guide.

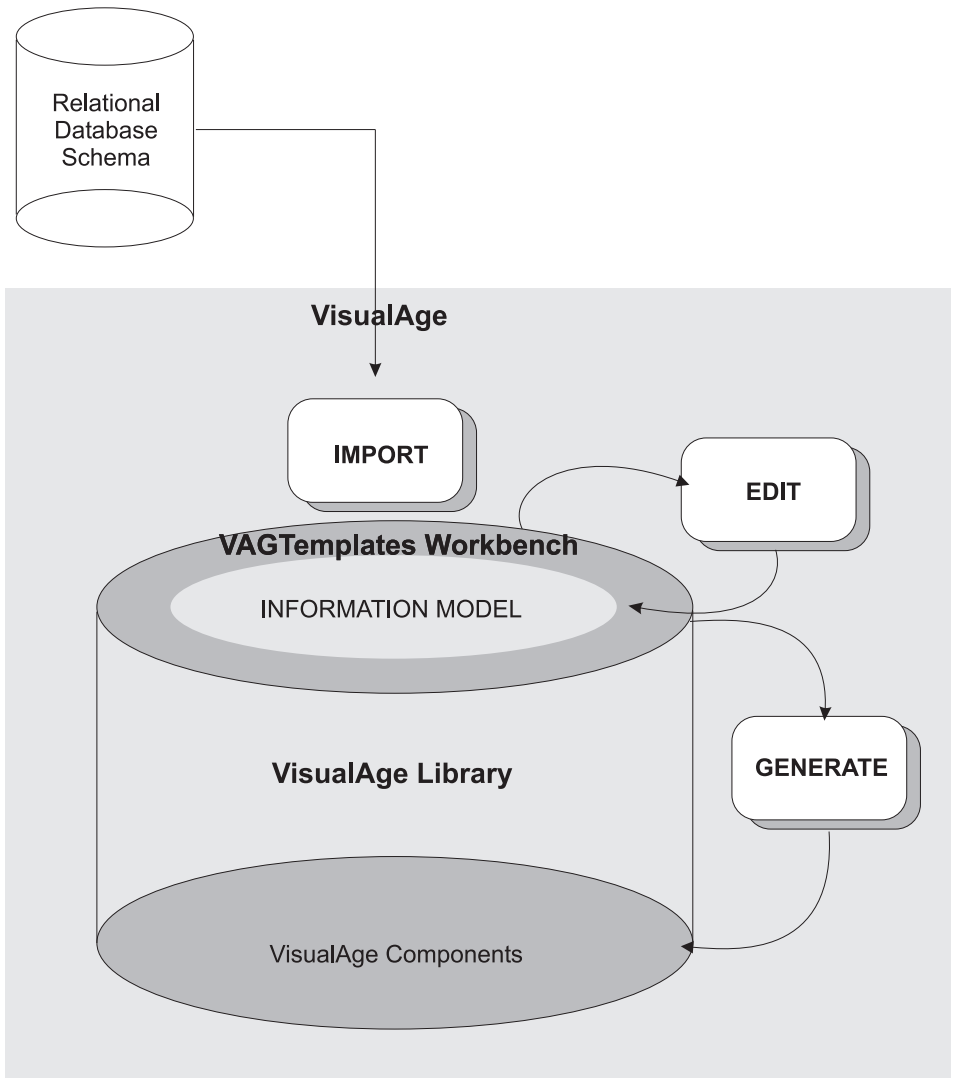


Figure 2. *VAGTemplates Overview*



---

## Chapter 3. Development Steps

The typical development effort with VAGTemplates consists of the following steps:

- *Defining a Workspace*

Each Workspace corresponds to one imported database and to generation parameters adapted to the application you want to generate. You define the generation parameters for your application in a given Workspace while the functional specifications for this application is not linked to this Workspace.

For more information on Workspace definition and parameterizing policy, see “Chapter 5. The Workspace” on page 15.

- *Importing a Relational Database*

The catalog of the existing relational database is read and its structure, comprising tables, views, columns and foreign keys, is imported into the VAGTemplates Information Model. This import creates pre-filled instances of the **Relational Table** and **Data Element** Information Model entities.

For more information on importing databases, see “Chapter 6. Relational Database Import” on page 19.

- *Defining Applications*

The VAGTemplates Workbench tools enable you to create, update, and delete instances of Information Model entities that you need to develop your application. These entities enable you to specify Web, GUI and TUI application components.

To develop your application, you need to:

- Complete the specification of imported **Data Elements**, when necessary; you can also create and specify new **Data Elements**.
- Specify **Value Styles** for the **Data Elements**, if necessary.
- Structure data into **Business Objects**. You need to define: the **Relational Tables** to which each **Business Object** is mapped, which of these table columns compose the **Business Object**, and which of the **Business Object** fields are used as *extract criteria* and *sort criteria*. You also specify the fields that will be used in detail layouts and those that will be used in list layouts.
- Specify **Interface Units**. Each **Interface Unit** must present a number of **Business Objects** and contain a number of child **Interface Units** as navigation targets.

For more information on defining applications see “Chapter 7. Application Definition” on page 23.

- ***Generating Applications***

The generators build components from logical specifications. The generators use default parameters or the parameters you set to build the application. The generated components are stored in packages/applications in the VisualAge repository.

VAGTemplates enables you to generate Web or GUI client and TUI components from the same functional description.

For more information on application generation see “Chapter 8. VAGTemplates Generation” on page 31.

- ***Enhancing the Applications***

The VAGTemplates-generated components are modular and include customizable components and components that have an API available for use with other components.

The VAGTemplates generators generate traceability information with each component. Use this information to determine which components, once generated, can be customized in the target environment, which components provide an API and which components are internal. This information is also used by the generators to preserve the customized components if you regenerate an application.

You can customize graphical components and insert 4GL, Smalltalk or Java code in special components designed to contain business logic code. These components, called *hooks*, are generated in the client and in the server.

- ***Customizing VAGTemplates***

VAGTemplates allows you to extend the entities of the Information Model by adding new specification fields.

In addition, it is possible to customize the VAGTemplates generators (in your VisualAge development environment) so that specific business standards are automatically taken into account in the generated components. New generators can also be developed.

For more information on customizing generators, see “Generator Customization” on page 36.

For more information on entity extension, see “Entity Extension” on page 35.



---

## Chapter 4. Presentation of the Workbench

The VAGTemplates Workbench enables you to import a database description into VAGTemplates, to model applications and generate Web, GUI and TUI client/server applications. The instances of the Information Model entities are stored in the VisualAge Library.

The main window of the Workbench is the Browser.

The **Action Bar** (below the menu bar) contains icons representing frequently performed actions in the Workbench. The icons have hover help to display action names.

The **Entities** area shows the Information Model entities you can manipulate when you are developing an application. You can select one or more entities from this list. The **Instances** area is displayed showing the instances that exist for the selected entities.

For detailed information on the Workbench, refer to the VAGTemplates Workbench Part, in your User's Guide.

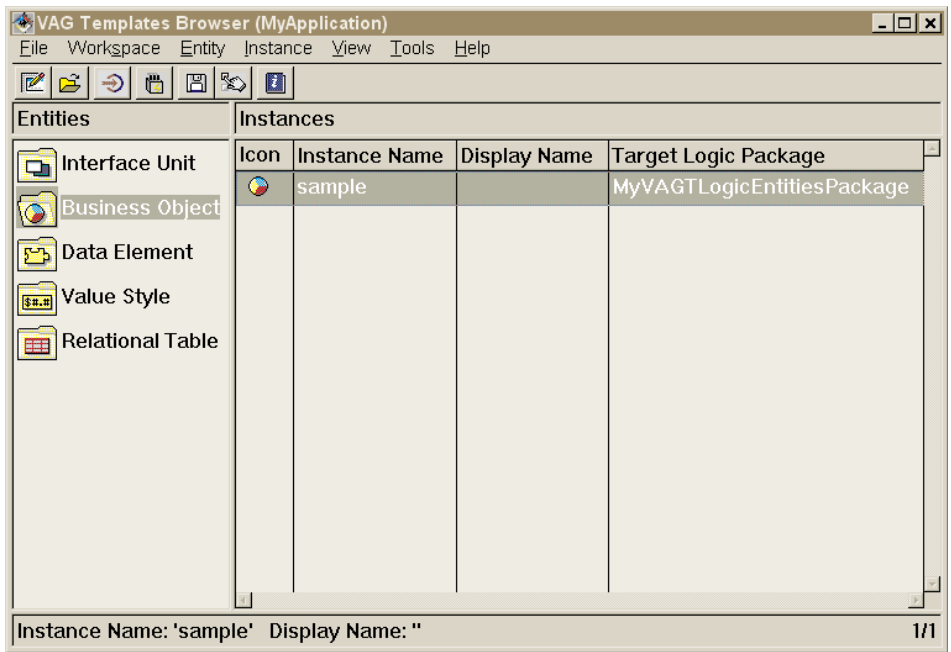


Figure 3. VAGTemplates Workbench

---

## Chapter 5. The Workspace

When you develop an application with VAGTemplates, you will need to create at least one Workspace. A Workspace definition is required to:

- **specify the generation parameters for the instances used by your application**

A Workspace is mainly used to gather a specific set of parameters required for generating a given application. If no Workspace is defined, entity or instance generation parameters cannot be accessed.

To generate an application specified with VAGTemplates you need to define a Workspace.

When you define a Workspace you specify a number of parameters that correspond to technical generation options and to the set of parameters that will apply to all the components generated from the Workspace.

A Workspace comprises:

- *Generation options* shared by *all* entities to be generated from the Workspace, for example the naming policy of the generated components
- *Default generation parameters* specified for each entity
- *Default generation parameters* specified for each instance
- *Parameter extensions* that can be defined as shared by all Workspaces or specific to a given Workspace

- **to import an existing relational database**

You also need to open a Workspace when importing a relational database. The import function can be accessed only if a Workspace is open since the description of one or more existing relational databases is always imported into the current Workspace.

For more information on Workspace definition, refer to “Part 2. Getting Started” on page 39 and to the *VAGTemplates User’s Guide*.

---

### Independence of Functional Description and Generation Parameters

To ease teamwork by providing more flexibility in specifications management, applications modeled in VAGTemplates are described by two principal elements that are independent: the functional or logical description and the parameter description.

The functional description is comprised of instances of model entities and their relationships. For a typical modeled application, it basically consists of a Business Object having data fields corresponding to Data Elements from one or more Relational Tables. The Business Object is called by an Interface Unit.

The parameter description is a set of generation parameters comprised of Workspace settings and entities parameters that are used to generate applications with different looks, behaviors and functions.

The functional description and the parameter description of a modeled application are independent: this means that a functional description of an application can be generated with more than one parameter description.

This independence results in the following:

- The functional aspect of the modeled application is completely independent of any Workspace, being only linked to the VisualAge image.
- For each entity instance, the functional description is unique while the generation parameters can be defined for and linked to several Workspaces. You have two types of editors for defining instances of each entity type. Use the **Definition** editor to define the functional description and the **Generation Parameters** editor to define the generation parameters.
- No Workspace definition is required to edit the functional description.
- For an application, there can be several Workspaces defined, each Workspace being a specific set of generation parameters.
- The application functional specifications can be stored in a VisualAge package or application independently of any Workspace.
- Each Workspace definition can be stored in its own package or application.
- Each set of generation parameters defined within a Workspace for a given instance can be stored in a package or application other than the one used to store its functional properties and/or the one used to store the Workspace definition.

---

## Setting Parameters

The VAGTemplates parameterizing policy enables you to standardize application development.

Default generation parameters are defined at the entity level for a particular application.

For example, for the **Business Object** entity, you can define default parameters by selecting the entity and *Default parameters* from the *Entity* drop-down menu.

For each entity instance, the functional properties that are unique, and the parameters that can be defined for each Workspace, can be separately edited and stored. This makes teamwork easier. Once the functional specifications are set up, it enables several developers teams to work simultaneously on the parameterizing of the same set of instances, using different workspaces.

When a new instance is defined, fields in its Generation Parameters editor panels are already filled in with the corresponding entity's default parameters. However, if different values are needed for a specific instance, they can be redefined at the instance level.

In the panels of the Generation Parameters editor of an instance, a *Redefined* checkbox is associated with each parameter, indicating if the parameter value is the default one or not. When you modify default values, *Redefined* is automatically checked. You can manually uncheck it, thus changing the current value to the default one (defined on the entity level), or check it even when the value is the default one, thus protecting the value against a change on the entity level.

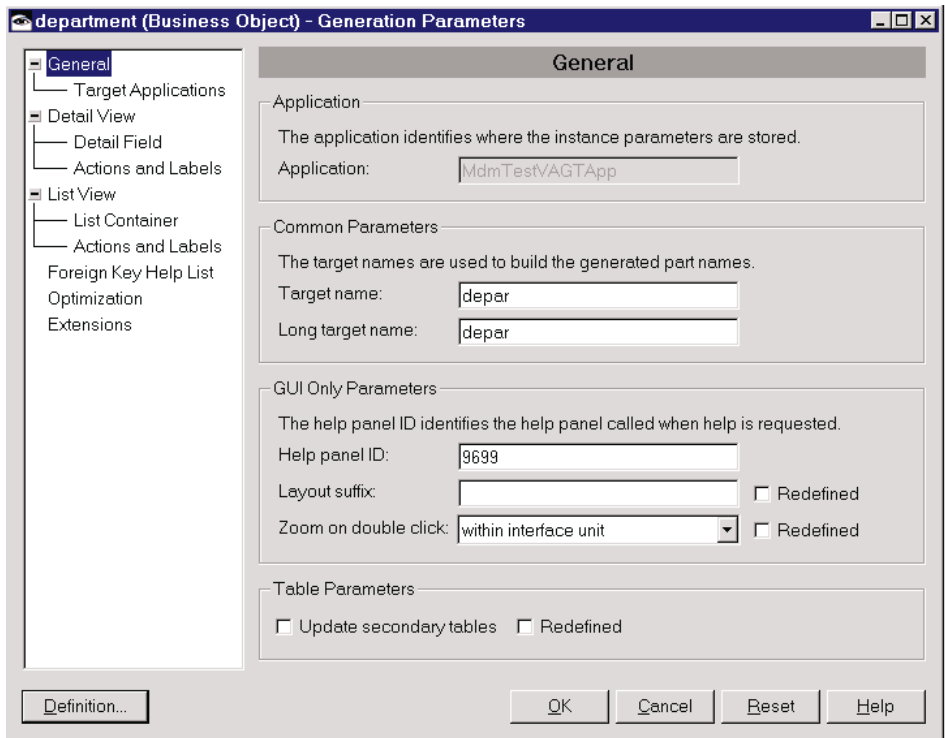


Figure 4. Business Object (sample) Generation Parameters editor: General Panel

**Tip:** Using default parameters to describe instances of VAGTemplates standard entities streamlines the development process. When you keep the default parameters, you only have to define logical descriptions in the Definition editor.

For more information on entity logical description, see “Chapter 7. Application Definition” on page 23.



---

## Chapter 6. Relational Database Import

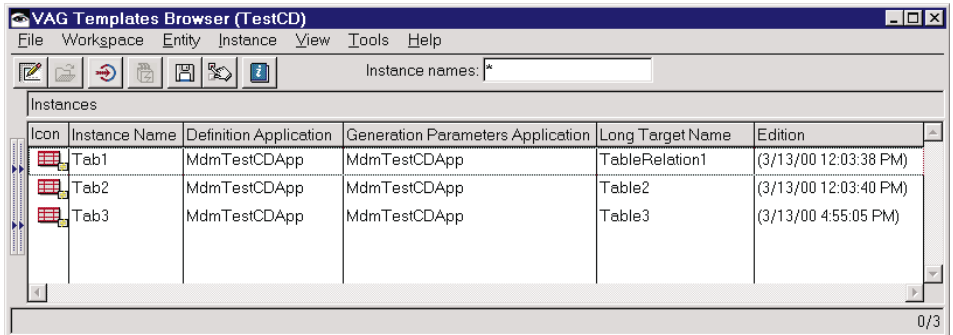
Using the import function, you can retrieve the description of a relational database to use in developing the application. The import into VAGTemplates automatically creates **Relational Table** and **Data Element** instances. DB2 and Oracle 7 database management systems are supported.

---

### Relational Table: Imported Data Storage

A **Relational Table** is either a table with columns, or a view with columns from one or more tables.

**Relational Table** instances are automatically created when you import the database. Each **Relational Table** instance describes the structure of a table from the imported relational database.



The screenshot shows the VAG Templates Browser (TestCD) interface. The main window displays a table of instances. The table has the following columns: Icon, Instance Name, Definition Application, Generation Parameters Application, Long Target Name, and Edition. There are three rows of data, each with a red icon representing a table.

Icon	Instance Name	Definition Application	Generation Parameters Application	Long Target Name	Edition
	Tab1	MdmTestCDApp	MdmTestCDApp	TableRelation1	(3/13/00 12:03:38 PM)
	Tab2	MdmTestCDApp	MdmTestCDApp	Table2	(3/13/00 12:03:40 PM)
	Tab3	MdmTestCDApp	MdmTestCDApp	Table3	(3/13/00 4:55:05 PM)

Figure 5. VAGTemplates Browser with Relational Table List of Instances

Use the Relational Table Definition editor to look at the definition of an imported **Relational Table** or to update the **Data Element** calls in the table columns.

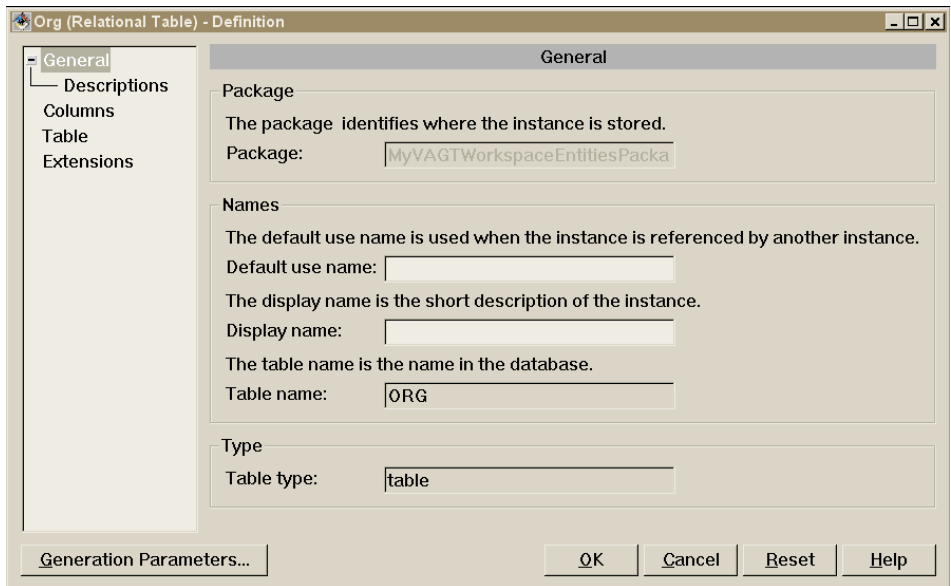


Figure 6. Relational Table (STAFF) Definition Editor: Genral Panel

For more information on the definition of a **Relational Table**, refer to the *VAGTemplates User's Guide*.

---

## Data Element: Imported Elementary Information

A **Data Element** represents elementary information stored in a **Relational Table** column or manipulated as a **Business Object** field. Its description includes a type, a length, labels, online help text for end users, value checks, etc.



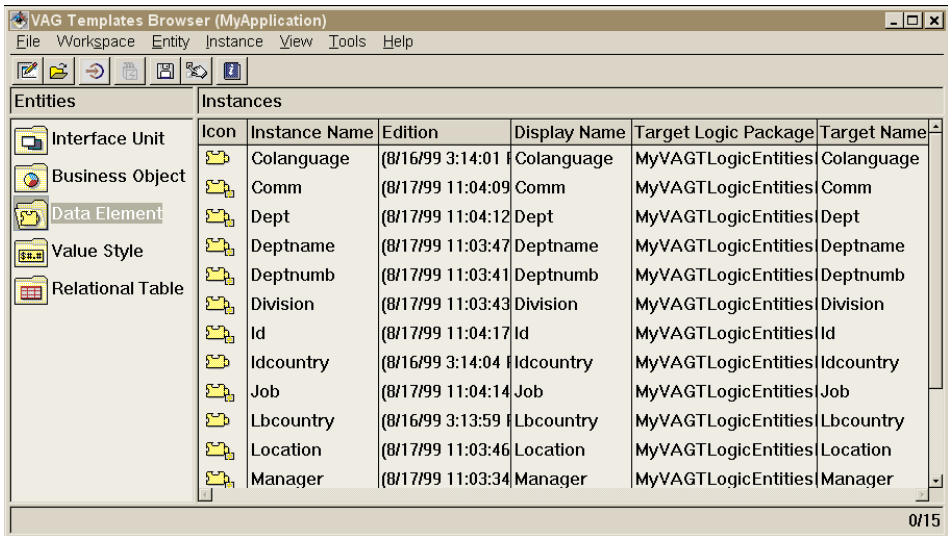


Figure 7. VAGTemplates Browser with Data Element List of Instances

VAGTemplates manages five **Data Element** types:

**Alphanumeric** Character string-type data

**Numeric** An integer or decimal, signed or unsigned numeric value

**Date** A date value with year, month, and day

**Time** A time value with hours, minutes, and seconds

**Timestamp** A concatenated date and time value used to manage versioning

**Data Element** instances are automatically created when you import the database.

If you select the *Re-Use Data Elements* check box in the Import from Database window at import time, VAGTemplates creates one instance of **Data Element** for all table columns that are identical in the database; otherwise it will create one instance of **Data Element** per column of the database tables.

You can complete the definition of an imported **Data Element** or create additional **Data Element** instances.

For more information on the Data Element editors used to update a **Data Element** pre-filled during the import or to define a **Data Element** that you created, see “Chapter 7. Application Definition” on page 23.



---

## Chapter 7. Application Definition

To describe your application, you need to create **Business Object** and **Interface Unit** instances through which data will be manipulated.

---

### Data Element: Description Completion

During the database import, a number of entry areas are filled in automatically.

The following entry areas are filled in with information from the database:

- **Display name**
- **Value type**
- **Format**

The following entry area is filled in by VAGTemplates:

- **Label**
- **Default Value Style**

You can enhance the description of the imported **Data Elements**. You can also create Data Elements instances.

Use the Data Element Definition and Generation Parameters editors to enhance a **Data Element** that has been pre-filled during the import or to define a **Data Element** that you created.

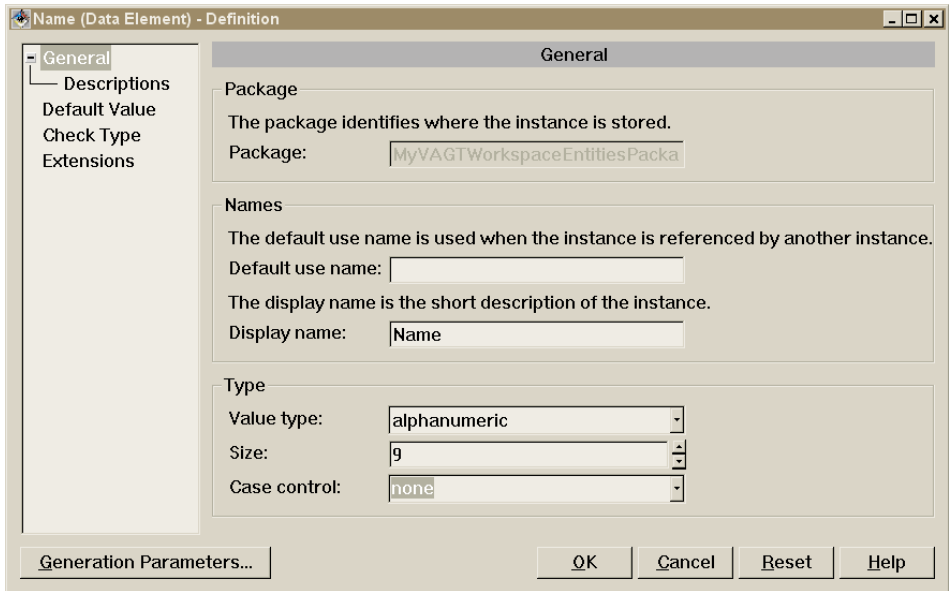


Figure 8. Data Element (NAME) Definition Editor: General Panel

Use the Definition editor panels to define information such as the type of the **Data Element** and the type of check applied to the data.

For more information on the **Value Style** entity, refer to “Value Style: Data Element Presentation Style”.

VAGTemplates provides the following types of checks to control incorrect data input:

- Check by value table: a list of authorized values is associated with elementary data so that the end user will not be allowed to enter a value that is not in the list.
- Check by interval: an interval of values is associated with elementary data so that the end-user will not be allowed to enter a value that is not in the interval.

For more information on the definition of a **Data Element**, refer to the *VAGTemplates User’s Guide*.

---

## Value Style: Data Element Presentation Style

A **Value Style** is a set of characteristics defining a presentation style for a numeric, a date, or a time value. It contains all the display and input characteristics that a numeric, date, or time value will have in the generated applications. One **Value Style** instance allows you to standardize the

presentation of several **Data Elements** of the same type. VAGTemplates manages four types of **Value Styles** (Numeric, Date, Time, Timestamp), one for each type of **Data Element** likely to have a **Value Style**.

A **Value Style** contains characteristics such as: sign symbol or unit symbol for a numeric **Data Element**, alignment or date formats for a date **Data Element**. For example, you can define a **Value Style** instance where the \$ sign is associated with the value and another instance where the £ sign is associated with the value.

Use the Value Style Definition editor to define a **Value Style**. No generation parameters can be specified for the Value Style entity.

The following **Value Style** specifications corresponds to a numeric value.

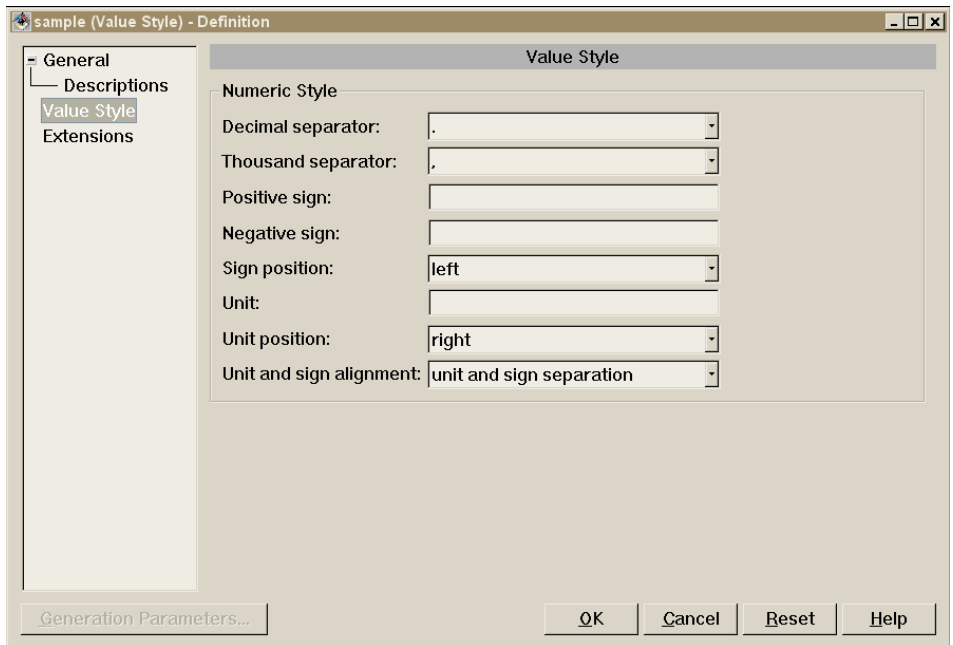


Figure 9. Numeric Value Style (sample) Definition Editor : Value Style Panel

For more information on the definition of a **Value Style**, refer to the *VAGTemplates User's Guide*.

## Business Object: Application Data View

A **Business Object** is a set of fields developed to match a specific application requirements for reading, presenting, and updating data. The fields of the **Business Object** represent **Data Element** that map to the columns of one or more **Relational Tables**.

For example, an application can use a Business Object called *Staff* with Data Elements that are *employee number*, *name*, *job*, and *salary* mapping to columns of the *Employee* and *Staff* Relational Tables.

A **Business Object** can also present non-persistent data using variable fields.

The **Business Object** maps to **Relational Tables**. Among these, the first table mapped is designated as the primary table of the **Business Object**. Only the primary table's fields can be updated; the fields of the other tables are read-only.

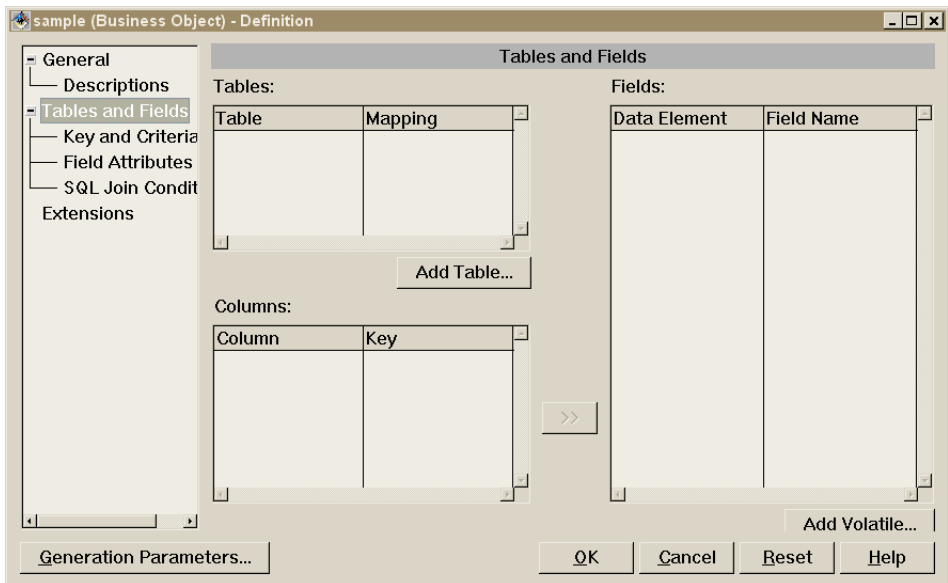


Figure 10. Business Object (sample) Definition Editor: Tables and Fields Panel

The presentation of fields in a **Business Object** depends on the characteristics defined for the **Data Elements** (label, value display, etc.), and on the **Business Object** presentation (detail or list).

To define a **Business Object**:

- Create a **Business Object**.

- Select the tables that the **Business Object** can access (primary table and others), and select its fields (**Data Element** calls).

Use the Business Object Definition and Generation Parameters editors to define a **Business Object**.

For more information on the definition of a **Business Object**, refer to the *VAGTemplates User's Guide*.

---

## Interface Unit: End User Interface

An **Interface Unit** is the user interface that presents the **Business Objects** and specifies whether a **Business Object** is used as a detail or as a list. The **Interface Unit** also defines navigation between **Interface Units**.

You will define an **Interface Unit** as a set of **Business Objects** and determine whether each **Business Object** should be presented as a detail or as a list:

- A detail **Business Object** is used to represent one row that can contain information from one or more tables
- A list **Business Object** is used to represent many rows with information from one or more tables

Each **Interface Unit** can display other **Interface Units**.

To define an **Interface Unit**:

- Create an **Interface Unit**.
- Indicate which **Business Objects** the **Interface Unit** contains and indicate for each **Business Object** whether it is used as a detail or as a list.
- Indicate the **Interface Units** which can be displayed.

Use the Interface Unit Definition and Generation Parameters editors to define an **Interface Unit**.

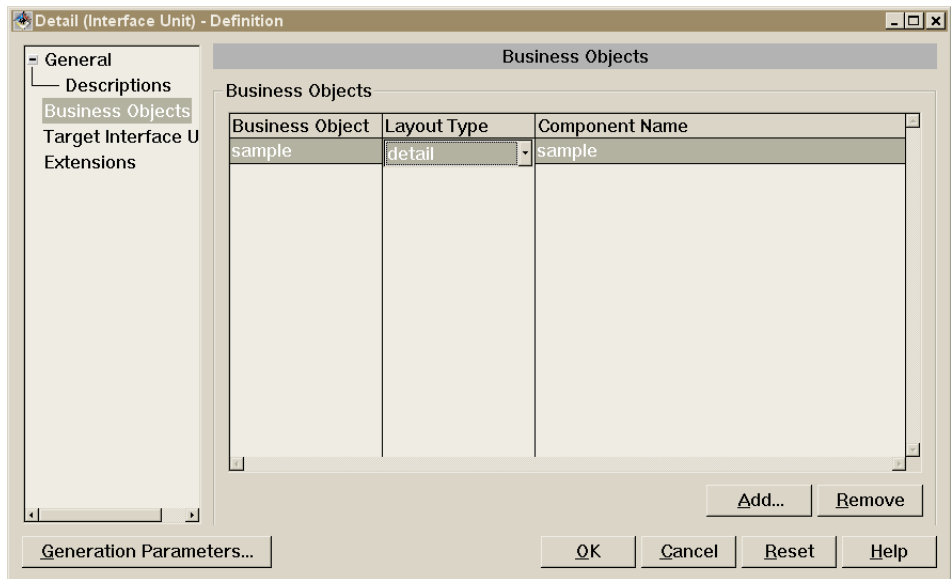


Figure 11. Interface Unit (Detail) Definition Editor: Business Objects Panel

For more information on the definition of an **Interface Unit**, refer to the *VAGTemplates User's Guide*.

## Help

Using VAGTemplates, you can provide the end user with online help for generated applications. Help text can be entered in the *Descriptions* panel in the Definition editor of the Data Element, Business Object, and Interface Unit entities. This online help description is the content that will be displayed when the end user requests help. Help generators allow you to generate help for GUI client applications and TUI applications.

- **On-line Help for GUI Client Applications**

Once the application is generated, each of its windows has a **Help** menu allowing the end user to display Help for the current panel. The same Help panel will be displayed if the F1 key is pressed when the focus is on the window. However, pressing F1 while the focus is in one of the **Interface Unit** fields displays the context-sensitive help panel for this field. The help panel displays the text you entered in the online help description field of the called **Data Element**.

An **Interface Unit** help panel contains general help for the window itself but also hypertext links to call Help for each **Business Object** contained in the **Interface Unit** and for each of its child **Interface Units**. A **Business Object** help panel contains general help for the **Business Object** itself and hypertext links to call help for each field in the **Business Object**.



- **Online Help for TUI Applications**

Once the application is generated, each of its maps includes a help key allowing the end user to display help for the current map. You can specify the presentation of the online help in the VAGTemplates Workbench:

- It can be displayed in a pop-up. In this case the online help text appears in a frame at the bottom of the current map.
- It can be displayed in a separate map. In this case the online help is displayed on the full screen.

In both cases, function keys are available in the help panel.

An **Interface Unit** help panel contains general help for the map itself concatenated to the help for each **Business Object** in the map, and concatenated to the help for each field in the **Business Object**. However, depending on the position of the cursor, the end user will also be able to request help directly on a field or on a set of fields.



---

## Chapter 8. VAGTemplates Generation

---

### Generation Overview

The VAGTemplates generators are activated from the VAGTemplates Workbench. From the specifications of your application, they produce VisualAge Generator parts and components specific to your VisualAge development environment. The generation process uses VAGTemplates entity instances as input.

**Note:** You can also generate components from a Workspace instance. The components generated from the Workspace generation are called *predefined parts/beans* and are shared by the applications generated from the Workspace. Error handling components are an example.

### VAGTemplates on Smalltalk 3.x Generators

VAGTemplates 4.0 provides you with a set of generators corresponding to Smalltalk 3.x generators. They are available with the current version of VAGTemplates but will become obsolete in VAGTemplates 4.1. Consequently, users who customized generators with VAGTemplates 3.x must migrate their customized generators to the 4.0 version of VAGTemplates.

The concerned generators are the following:

- 4GL GUI generators
- Smalltalk GUI generators
- TUI generators

**WARNING:** New VAGTemplates developers must NOT use these generators.

**Note:** For documentation on the migration tool, refer to the *VisualAge Generator Migration Guide*.

### VAGTemplates on Smalltalk Generators

VAGTemplates on Smalltalk provides you with four sets of generators:

- Smalltalk GUI generators to generate the whole GUI client application including **Business Object** and **Interface Unit** layouts.
- Smalltalk TUI generators to generate the whole TUI application including **Business Objects** and **Interface Unit** layouts (organized into maps and map groups).
- Smalltalk Web generators to generate the whole Web-based application.

- Smalltalk Help GUI generators to generate online help for GUI client applications. Help is generated into RTF and HPJ files for Windows, and IPF files for OS/2.

## VAGTemplates on Java Generators

VAGTemplates on Java provides you with three sets of generators:

- Java GUI generators to generate the whole GUI client application including **Business Object** and **Interface Unit** layouts.
- Java TUI generators to generate the whole TUI application including **Business Objects** and **Interface Unit** layouts (organized into maps and map groups).
- Java Web generators to generate the whole Web-based application.

---

## Generated Beans/Parts

The generated beans/parts present a modular structure. The generated beans/parts that can be enhanced in the target tool are independent from the beans/parts providing an API which must not be modified outside VAGTemplates.

- *Customizable beans/parts*

The following are some of the customizable beans/parts.

- **Layouts**

Layouts are automatically generated by VAGTemplates. Once generated, you can enhance them and maintain their description at the target tool level.

For example, for a GUI **Business Object**, you will generate a **Business Object** layout bean/part containing the graphical elements of the **Business Object**. For a TUI **Interface Unit**, you will generate a header map, a trailer map, etc.

- **Hooks**

Hooks are insertion points generated for you to add business logic. Hooks are called during the execution of the application, even if they contain no code.

For example, if you specified a customized check for a **Data Element**, a hook will be generated allowing you to add an algorithmic check. This hook will be executed when data is entered in the corresponding input area.

- *Beans/Parts providing an API*

These parts provide you with an API to manage data accesses, data checks, etc. This API is used to easily enhance the customizable components in the target environment. To modify these components' behavior, you have to modify the description of the corresponding instances in VAGTemplates and regenerate them.

For example, for a **Business Object**, you will generate a **Business Object** bean/part offering access services such as data input check, instance creation, read, update, and deletion.

**Note:** Both types of beans/parts rely on additional internal components that should not be modified.



---

## Chapter 9. Generator Customization

It is possible to customize generators using the loadable feature called **VAGTemplates with Customizing Support** (available with a separate license).

---

### Entity Extension

If entities in the VAGTemplates Information Model do not cover all your specification needs, you can extend them by adding specification fields. The VAGTemplates Workbench allows you to enhance the existing Information Model by providing you with extension panels for each entity type. The definition as well as the generation parameters of each entity type can be extended. You define extensions at the entity level and they will be stored in the Information Model.

Use the extension panels to extend an entity (*Entity* menu / *Extend Definition* or *Extend Generation Parameters* choices).

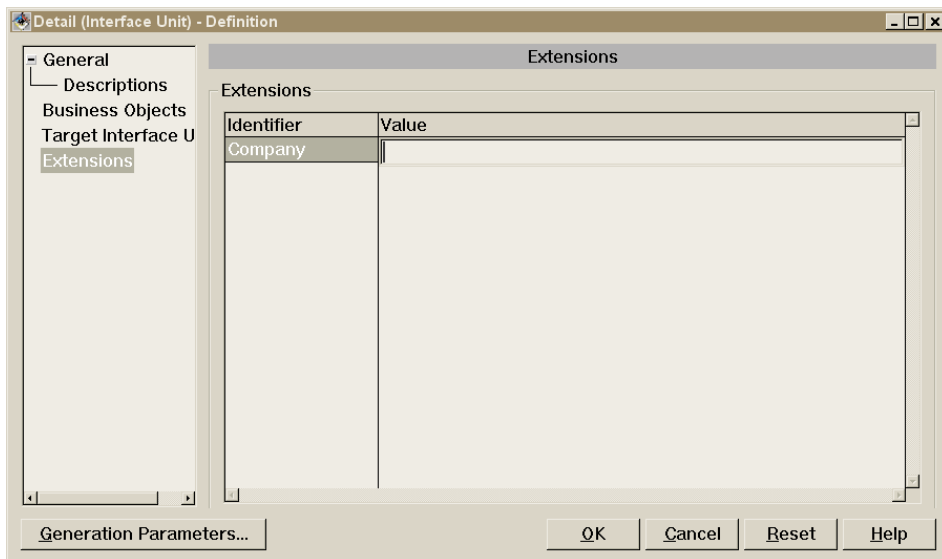


Figure 12. Interface Unit Definition Editor: Extensions Panel

You extend an entity by defining additional specification fields. The values of these fields can be alphanumeric, numeric, date, time or timestamp.

When an entity is extended, all the additional specification fields are transferred to the instance level. They are respectively displayed in the *Extensions* panel of the Definition editor or Generation Parameters editor.

For more information on entity extension, refer to the *VAGTemplates User's Guide*.

To make full use of the extensions you will have to customize the standard generators. This is easily done using the API that VAGTemplates offers to manipulate the extensions.

For more information on generator customization, see “Generator Customization”.

---

## Generator Customization

VAGTemplates provides facilities for either customizing the standard generators or creating new generators, in your target environment.

A generator is implemented as a Java or Smalltalk class that inherits services from a generation framework.

To customize a generator you create a Java or Smalltalk class that inherits from the class of the standard generator you want to customize, and redefine the standard methods that need to be changed. Generator customization is necessary to make use of Information Model entities that have been extended. You might also want to customize the generators for other specific generation needs not related to extended Information Model entities.

For more information on entity extension, see “Entity Extension” on page 35.

To create a new generator you create a Java or Smalltalk class that inherits from a generator class. These new generators can use the VAGTemplates API that provides general services for the generation of VisualAge Generator parts and VisualAge for Java or VisualAge Smalltalk Enterprise components, and for accessing entity specifications.



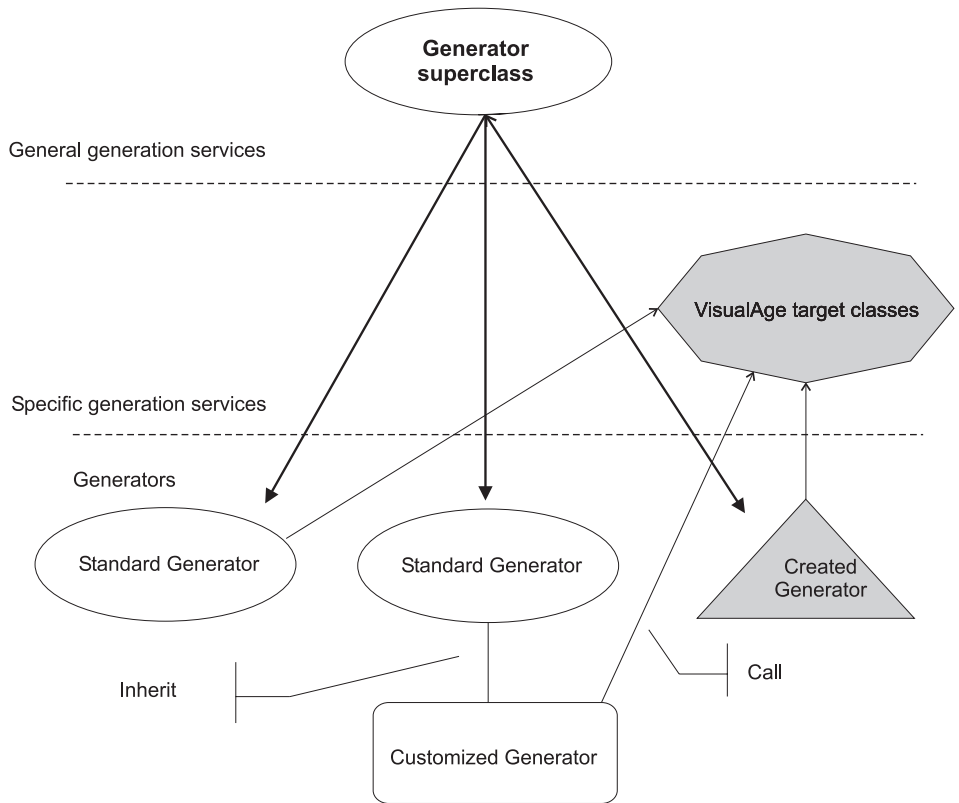


Figure 13. Generators Design

For more information on the generators, refer to the *VAGTemplates Customizer User's Guide*.

Examples of generator customization are also given in the *VAGTemplates Customizer User's Guide*.



---

## Part 2. Getting Started



---

## Chapter 10. Introduction

This Part gets you started using VAGTemplates and guides you through building your first simple application. This example is built using the **VAGTemplates on Smalltalk** development environment. The procedure to develop this application will be generally the same for a GUI client application and for a TUI application.

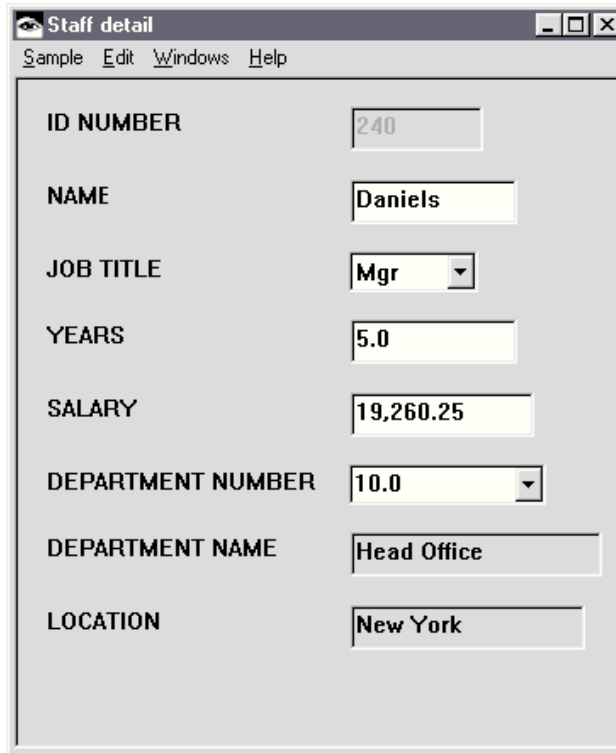
You will create an application with a single window (GUI) or map (TUI) which displays a detail allowing the end user to visualize information about a staff member.

This application will use the STAFF and ORG tables of the DB2 SAMPLE database that comes with DB2.

For more details on the DB2 SAMPLE tables, see your IBM Database Documentation.

Once your application is generated, you will be able to test it with the VisualAge Smalltalk Enterprise workstation.

The "Staff Details" GUI client application will look like this:



The "Staff Details" TUI application will look like this :

```

SYST: DS2CICS          HYAPPLICATION          FASTPATH: Detail
USER: PC user         Staff Details          05-26-1997  10:11:24

ACTIONS: I INSERT U UPDATE D DELETE
ACTION: _
NAME Daniels
ID          240
DEPTNUMB   10
JOB TITLE  Mgr
YEARS      5
SALARY     19260
DEPARTMENT Head Office
LOCATION     New York

FASTPATH: █ Daniels

PF01: HELP  PF02:          PF03: EXIT  PF04: LOOKUP PF05:          PF06:
PF07:          PF08:          PF09:          PF10:          PF11:          PF12: CANCEL

```

This chapter explains how to do the following:

- Start VAGTemplates and define a new Workspace,
- Import a relational database schema,
- Explore the pre-filled instances of entities after import,
- Create a Business Object,
- Create an Interface Unit presenting this Business Object,
- **TUI** Create a root Interface Unit calling this Interface Unit,
- Generate the application,
- Test your application with the VisualAge Smalltalk Enterprise workstation.

**Note:** NOTE:

The VAGTemplates user interface design matches the look of the VisualAge family products in general, and especially with VisualAge Generator. The interface clarifies the function of the Information Model entity properties.





---

## Chapter 11. Starting VAGTemplates and Defining a New Workspace

To start VAGTemplates:

- Open your VisualAge Smalltalk Enterprise workstation and select the *Open VAGTemplates Browser* choice from the *VAGT Tools* menu in the Organizer's *Tools* menu.

The first time you open VAGTemplates, the *Select Workspace* window opens up. If you want to prevent this window from being automatically displayed later on, disable the *Show this window at startup* option. This will directly open the VAGTemplates Browser.

To define a new Workspace:

1. If the *Select Workspace* window opens when you start VAGTemplates, check *Create a new Workspace* in this window.
2. Or, if the Browser opens when you start VAGTemplates,
  - from the *Workspace* menu, select *Open Workspace*,
  - In the *Open Workspace* window, check *Create a new Workspace*.
3. Enter the name of the Workspace you want to create.  
For this sample application, enter *MyApplication* in the *Workspace name* field.
4. Specify an application where the new Workspace will be stored.  
For this sample application, enter *MdmExample1App* in the *Application* field.
5. Click *OK*.

**Note:** You can choose not to create a Workspace by checking the *Do not create a workspace* option. In this case, you would only be able to define the functional specifications for your application, the generation parameters being not accessible. Also, keep in mind that the definition of a Workspace is required for both application generation and the import of the database schema.

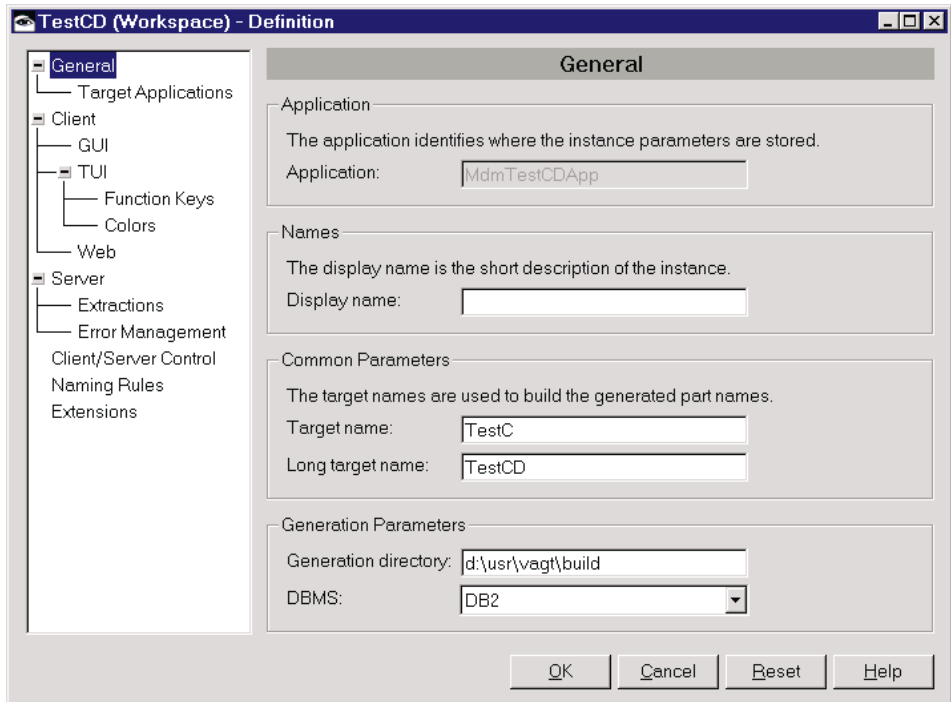


Figure 14. Workspace Definition Editor: General Panel

**Note:** The **Mdm** prefix for specification applications is a convention we use to differentiate these applications from the product applications, which are prefixed with **Mdl**.

To define the Workspace *MyApplication*:

1. Select *Definition* from the *Workspace* drop-down menu.  
The Workspace Definition editor opens displaying the *General* panel.
2. The *Application* field indicates the name of the application associated with the Workspace.
3. In the *Display name* field, enter a display name for the Workspace. It will be displayed in the list of called instances in the generated application help panels.
4. The *Target name* input field allows you to enter a character string that will be used to build the name of the components that are generated from the Workspace entity. By default it is the first five characters of the Workspace's name, but it is possible to enter up to 64 characters in the *Long Target Name* field, if needed.

5. The *Generation directory* input field displays the default directory where the generated files will be stored (e.g. help file). Enter a new directory path if you want these files to be stored elsewhere.
6. In the *Target Applications* panel, the three edit fields indicate the default application where the generated components will be stored. You can change it here if you want.
7. **TUI** In the *Client/Server Control* panel, select *Server* from the *Control location* drop-down list in order that the checks we will define for the Data Elements can be implemented.
8. Click *OK*.

For more information on the Workspace settings, refer to the Part dedicated to the VAGTemplates WorkStation, in your User's Guide.

For information on the predefined Parts, refer to the Standard Use of VAGTemplates Part, in your User's Guide.

For information on check implementation in GUI and TUI client applications, refer to the Standard Use of VAGTemplates Part, in your User's Guide.



---

## Chapter 12. Importing a Relational Database

To import an existing relational database :

1. Start DB2
2. In your VAGTemplates Browser, open the Workspace in which you want to import the database  
For this sample application, open the *MyApplication* Workspace.
3. Select the **Import from Database** choice from the *Tools* drop-down menu.  
The *Import from Database* window opens.
4. If the *Database name* field does not contain the name of the database you want to import:
  - Click the *Change* button.  
The *Settings* window opens up.
  - Select the database you want to import.  
For our sample application, select SAMPLE.
  - Click *OK*.
5. The *Target application* field indicates the name of the application where the imported instances of Data Elements and Relational Tables will be stored. By default, the application is that of the current Workspace — *MdmExample1App* for this sample application.

**Note:** The imported instances can be moved to other applications from the VAGTemplates Browser.

6. Select the tables you want to import:
  - In the *Search Criteria* area, check the *Tables* option.
  - Click the *Build List* push-button.
  - In the list of available tables, select the STAFF and ORG tables.
  - Click the >> button.
7. In the *Import from database* window, select the *Reuse data elements* check box.
  - If this check box is not selected, VAGTemplates creates a unique Data Element per column even if two columns have the same name and the same format.
  - If this check box is selected, only one Data Element will be created.  
*For example, if there are two NAME columns with the same format, the import will only create one NAME Data Element.*  
However, if two columns have the same name (*Name*) but have different formats, two Data Elements are created (*Name* and *Name1*).

8. Click the *Import* push-button to activate the import.

For more details on the *Import from database* window, refer to the VAGTemplates Browser Part, in your User's Guide.

For information on parameterizing DB2 database for import, refer to "Parameterizing DB2 Database Import" on page 91.

---

## Chapter 13. Exploring the Pre-filled Instances

The import has created several Information Model entity instances.

- *Pre-filled Relational Table instances*

VAGTemplates has imported the STAFF and ORG tables from the SAMPLE database which will be of particular interest in building our application.

The imported instances are stored by default in the same application as that of the Workspace.

The composition of the SAMPLE tables depends on the DB2 version you have. The following is one possible composition of the STAFF and ORG tables:

*Table 1. STAFF TABLE*

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
10	Sanders	20	Mgr	7	18357.50	-
20	Pernal	20	Sales	8	18171.25	612.45
30	Marenghi	38	Mgr	5	17506.75	-
40	O'Brien	38	Sales	6	18006.00	846.55
50	Hanes	15	Mgr	10	20659.80	-
60	Qwingley	38	Sales	-	16808.30	650.25
70	Rothman	15	Sales	7	16502.83	1152.00
80	James	20	Clerk	-	13504.60	128.20
90	Koonitz	42	Sales	6	18001.75	1386.70
100	Plotz	42	Mgr	7	18352.80	-
110	Ngan	15	Clerk	5	12508.20	206.60
120	Naughton	38	Clerk	-	12954.75	180.00
130	Yamaguchi	42	Clerk	6	10505.90	75.60
140	Fraye	51	Mgr	6	21150.00	-
150	Williams	51	Sales	6	19456.50	637.65
160	Molinare	10	Mgr	7	22959.20	-
170	Kernisch	15	Clerk	4	12258.50	110.10
180	Abrahams	38	Clerk	3	12009.75	236.50
190	Sneider	20	Clerk	8	14252.75	126.50
200	Scoutten	42	Clerk	-	11508.60	84.20

Table 1. STAFF TABLE (continued)

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
210	Lu	10	Mgr	10	20010.00	-
220	Smith	51	Sales	7	17654.50	992.80
230	Lundquist	51	Clerk	3	13369.80	189.65
240	Daniels	10	Mgr	5	19260.25	-
250	Wheeler	51	Clerk	6	14460.00	513.30
260	Jones	10	Mgr	12	21234.00	-
270	Lea	66	Mgr	9	18555.50	-
280	Wilson	66	Sales	9	18674.50	811.50
290	Quill	84	Mgr	10	19818.00	-
300	Davis	84	Sales	5	15454.50	806.10
310	Graham	66	Sales	13	21000.00	200.30
320	Gonzales	66	Sales	4	16858.20	844.00
330	Burke	66	Clerk	1	10988.00	55.50
340	Edwards	84	Sales	7	17844.00	1285.00
350	Gafney	84	Clerk	5	13030.50	188.00

Table 2. ORG TABLE

DEPTNUMB	DEPTNAME	MANAGER	DIVISION	LOCATION
10	Head Office	160	Corporate	New York
15	New England	50	Eastern	Boston
20	Mid Atlantic	10	Eastern	Washington
38	South Atlantic	30	Eastern	Atlanta
42	Great Lakes	100	Midwest	Chicago
51	Plains	140	Midwest	Dallas
66	Pacific	270	Western	San Fransisco
84	Mountain	290	Western	Denver

- **Pre-filled Data Element instances**

VAGTemplates has automatically created several Data Element instances corresponding to the columns of the imported table.

These instances are stored by default in the same application as the Workspace.

For information on the Data Element creation policy, refer to the description of the above-mentioned topic *Reuse data elements*.



**TIP:** To see the list of the instances created by the import, click on each entity in the *Entities* area. The *Instances* area will display the names of the corresponding instances.

In our example, among all the imported Data Elements, the following ones will be of particular interest in building our application:

- ID
- NAME
- DEPT
- JOB
- YEARS
- SALARY
- DEPTNAME
- LOCATION

Now that you have created your Workspace, imported the database, and are aware of the instances that have been automatically created, you are ready to develop your application.



---

## Chapter 14. Defining the Data Element Labels and Input Checks

In this section, you will learn how to complete the description of the imported Data Elements. You will add default use names, display names, and on-line help text, and change the labels and add input value checks for some Data Elements.

For each Data Element, we can modify its functional description using the Definition editor, then when necessary, its parameters using the Generation Parameters editor.

To complete the functional description of the YEARS Data Element:

1. In the VAGTemplates Browser, from the list of entities, select the Data Element entity and from the list of instances, select the YEARS Data Element.
2. Right click and select *Definition* from the pop-up menu.  
The Definition editor opens for the selected Data Element.
3. In the *General* panel,
  - enter *Years* in the *Default use name* field.  
This use name will be used as the title of the help panel for this Data Element in the final application.
4. Open the *Descriptions* panel by selecting *Descriptions* in the tree view,
  - enter *Years is a smallint* in the *Textual Description* field.  
This description is used only to specify internal information that may be useful for other developers using the same instance.
  - enter *Enter an integer value ranging from 1 to 100* in the *On line help Description* field.  
This description is the help text that will be displayed in the help panel for this Data Element instance in the end-user application.
5. From the editor tree view, select *Default value*,
  - In the *Default Value* panel, select *constant* from the *Default value mode* drop-down list in order to define a default value for the YEARS Data Element. This default value will appear in the field when the detail opens.  
enter *1* in the *Value* field.
6. From the tree view, select *Check type*,

- From the *Check Type* panel, select *interval* from the *Check type* drop-down list.

The end user will be allowed to enter only authorized values.

- Select *closed-closed* in the *Type* drop-down list,
- Enter *1* in the *Minimum* field,
- Enter *100* in the *Maximum* field,
- Click *OK* to save the functional properties specified and close the Definition editor.

To define some generation parameters (labels in this sample application) for the YEARS Data Element:

1. In the VAGTemplates Browser, from the list of entities, select the Data Element entity and from the list of instances, select the YEARS Data Element.
2. Right click and select ***Generation Parameters*** from the pop-up menu. The Generation Parameters editor opens for the selected Data Element.
3. From the editor tree view, select *Labels*.
  - From the *Labels* panel, enter *Years* in the *Default label* field. This label will be displayed next to the field when presented in a detail.
  - enter *Years* in the *Column label* field. This label will be displayed as column header when the field is displayed in a list.
  - Click *OK* to save the parameters specified and close the editor.

To complete the functional description of the SALARY Data Element:

1. In the VAGTemplates Browser, from the list of entities, select the Data Element entity and from the list of instances, select the SALARY Data Element.
2. Right click and select ***Definition*** from the pop-up menu. The Definition editor opens for the selected Data Element.
3. In the *General* panel,
  - enter *Salary* in the *Default use name* field.
4. Open the *Descriptions* panel by selecting *Descriptions* from the tree view.
  - Enter *Salary is a dec (7,2)* in the *Textual description* field.
  - Enter *Enter a decimal value with a 7-digit capacity and a 2-digit precision.* in the *On line help description* field.
  - Click *OK* to save the properties specified and close the editor.

To define the SALARY Data Element labels:

1. In the VAGTemplates Browser, from the list of entities, select the Data Element entity and from the list of instances, select the SALARY Data Element.
2. Righth click and select *Generation Parameters* from the pop-up menu. The Generation Parameters editor opens.
3. From the editor tree view, select *Labels*.
  - Fom the *Labels* panel, enter *Salary* in the *Default label* field.
  - Enter *Salary* in the *Column label* field.
  - Click *OK* to save the parameters specified and close the editor.

To complete the functional description of the NAME Data Element and specify its labels:

1. In the *General* panel of the Definition editor,
  - enter *Name* in the *Default use name* field.
2. In the *Descriptions* panel of the Definition editor,
  - enter *Name is a varchar (9)* in the *Textual description* field.
  - enter *Enter a string value with 9 characters maximum.* in the *On line help description* field.
3. Click *OK* to save the properties specified and close the editor.
4. In the *Labels* panel of the Generation Parameters editor,
  - Enter *Name* in the *Default label* field.
  - Enter *Salary* in the *Column label* field.
5. Click *OK* to save the parameters specified and close the editor.

To complete the functional description of the LOCATION Data Element and specify its labels:

1. In the *General* panel of the Definition editor,
  - enter *Location* in the *Default use name* field.
2. In the *Descriptions* panel of the Definition editor,
  - enter *Location is a varchar (13)* in the *Textual description* field.
  - enter *Enter a string value with 13 characters maximum.* in the *On line help description* field.
  - Click *OK* to save the properties specified and close the editor.
3. In the *Labels* panel of the Generation Parameters editor,
  - enter *Location* in the *Default label* field.
  - enter *Loc* in the *Column label* field.
4. Click *OK* to save the properties specified and close the editor.

To complete the functional description of the JOB Data Element and specify its labels:

1. In the *General* panel of the Definition editor,
  - enter *Job* in the *Default use name* field.

2. In the *Descriptions* panel of the Definition editor,
  - enter *Job is a char (5)* in the *Textual description* field.
  - enter *Enter a string value with 5 characters maximum.* in the *On line help description* field.
3. In the *Default value* panel of the Definition editor,
  - select *constant* from the *Default value mode* drop-down list to define a default value for the Job Data Element.
  - enter *Clerk* in the *Value* field and *Office employee* in the *Comment* field,
4. In the *Check type* panel of the Definition editor,
  - select *value table* from the *Check type* drop-down list.  
The end user will be allowed to select only one of the proposed authorized values.
  - To specify the contents of the value table:
    - click *Add* and enter *Mgr* in the *Value* column and *Manager* in the *Comment* column,
    - click *Add* and enter *Clerk* in the *Value* column and *Office employee* in the *Comment* column. Continue for the following jobs:
      - Sales/Itinerant employee
      - Mkt/Marketing
  - Click *OK* to save the properties specified on the various panels and close the editor.
5. In the *Labels* panel of the Generation Parameters editor,
  - enter *Job title* in the *Default label* field.
  - enter *Job* in the *Column label* field.
6. **GUI** In the *Display* panel of the Generation Parameters editor,
  - select *drop-down list* for the *Value display* parameter.  
The list of values will be displayed in a drop-down list so that the end user can choose a value.
  - Click *OK* to save the parameters and close the editor.

To complete the functional description of the ID Data Element and specify its labels:

1. In the *General* panel of the Definition editor,
  - enter *IdNo* in the *Default use name* field.
2. In the *Descriptions* panel of the Definition editor,
  - enter *IdNo is a smallint not null* in the *Textual description* field.
  - enter *Enter an integer value different from 0* in the *On line help description* field.
  - Click *OK* to save the properties specified and close the editor.
3. In the *Labels* panel of the Generation Parameters editor,
  - enter *Id Number* in the *Default label* field.
  - enter *Id No* in the *Column label* field.
4. Click *OK* to save the properties specified and close the editor.

To complete the functional description of the DEPTNAME Data Element and specify its labels:

1. In the *General* panel of the Definition editor,
  - enter *DeptName* in the *Default use name* field.
2. In the *Descriptions* panel of the Definition editor,
  - enter *DeptName is a varchar (14)* in the *Textual description* field.
  - enter *Enter a string value with 14 characters maximum* in the *On line help description* field.
  - Click *OK* to save the properties specified and close the editor.
3. In the *Labels* panel of the Generation Parameters editor,
  - enter *Department Name* in the *Default label* field.
  - enter *Dept Name* in the *Column label* field.
4. Click *OK* to save the properties specified and close the editor.

To complete the functional description of the DEPT Data Element and specify its labels:

1. In the *General* panel of the Definition editor,
  - enter *Dept* in the *Default use name* field.
2. In the *Descriptions* panel of the Definition editor,
  - enter *Dept is a smallint* in the *Textual description* field.
  - enter *Enter an integer value* in the *On line help description* field.
3. In the *Default Value* panel of the Definition editor,
  - select *constant* from the *Default value mode* drop-down list in order to define a default value for the Dept Data Element. This value is 0 by default as this field is numeric.
  - Click *OK* to save the properties specified and close the editor.
4. In the *Labels* panel of the Generation Parameters editor,
  - enter *Department Number* in the *Default label* field.
  - enter *Dept No* in the *Column label* field.
5. Click *OK* to save the parameters specified and close the editor.





---

## Chapter 15. Creating a Business Object

To present the Data Elements in the final application, you need to create a Business Object that will call the Data Elements that must be displayed. This Business Object will present the Data Elements as fields and will provide actions to manipulate the data in these fields.

To create the Business Object:

1. From the VAGTemplates Browser, select *New...* from the *Instance* drop-down menu (or right click and select *New...* from the pop-up menu). The *New VAGT Instance* window opens.
2. Enter *Sample* in the *Instance name* field, and select an application where the Business Object will be stored in the *Application* combo box.

For this sample application select *MdmExample1App*, the application you specified when you defined the *MyApplication* Workspace and imported the SAMPLE database.

**Note:** This application is intended to store the functional description of this Business Object instance. You can input the name of a new application which will then be created when you create the Business Object. Different users can create their instances in applications that they manage.

3. In the *Instance type* area, select the Business Object entity.
4. Select the *Open now* check box to open the Business Object Definition editor as the creation ends.
5. Click *OK*.



---

## Chapter 16. Defining a Business Object

---

### Definition

The Business Object Definition editor enables you to specify the functional aspects of your Business Object. Use the editor tree view to navigate from one panel to another.

*General panel:*

1. Enter *Sample* in the *Default use name* field.
2. In the *Display name* field also enter *Sample*.

*Descriptions panel:*

1. In the *Textual Description* field type *Business Object used to show Staff*.
2. In the *On line help description* field:
  - If you are developing a GUI client application, enter for this example: *This Business Object is used to consult, modify or delete information of a staff member. Enter the id of the staff member you want to see and select Read from the Sample menu.*
  - If you are developing a TUI application, enter for this example: *This Business Object is used to consult, modify or delete information of a staff member. Enter the id of the staff member you want to see in the access parameters area and press enter.*

*Tables and Fields panel:*

1. Define the first table called by the Business Object.
  - Right click in the *Tables* list, select **Add** from the pop-up menu, or click the *Add* push-button.
  - Select STAFF in the list from the *Add Tables* dialog that opens.
  - Click OK.
  - Define the fields of the Business Object that correspond to columns of the STAFF table.
    - Select STAFF from the *Tables* list.

**Note:** The first table called is the primary table of the Sample Business Object. In the final application, the end user will be authorized to update the data from this primary table.

- Select ID from the *Columns* list; hold down left mouse button and drag the mouse over NAME, DEPT, JOB, YEARS and SALARY to select them (you can also press the CTRL key and select the fields).

- Click the >> button.
- Define the other tables called by the Business Object.
  - Right click in the *Tables* list, select **Add** from the pop-up menu, or click the *Add* push-button.
  - Select ORG in the list from the *Add Tables* dialog that opens.
  - Click *OK*.
- Define the fields of the Business Object that correspond to columns of the ORG table.
  - Select ORG from the *Tables* list.
  - Press CTRL and select DEPTNAME and LOCATION from the *Columns* list.
  - Click the >> button.

**Note:** In the final application, the data corresponding to the secondary tables will be read-only.

*Key and Criteria* panel:

- **Logical key**

The logical key is required to successfully perform CRUD actions on fields (creation, update, etc.) in the final application. The logical key consists of one or more of the Business Object's fields. It must be unique. One value of the logical key is used to select one and only one occurrence of the Business Object from the database.

To define the logical key of the Business Object:

1. Select *IDNo* from the *Fields* list,
2. Click the corresponding >> button.

- **Extract criteria**

Extract criteria are used to filter the data accessed from the database. An extract criterion consists of one or more fields.

To define the extract criteria for the Business Object:

1. Select *LOCATION* from the *Fields* list,
2. Click the corresponding >> button.
3. Click in the *Retrieve policy* column and select *match* from the drop-down list.

With the operator *match*, the end user will enter a character string in the extract criteria field (in this case, a location) and the rows extracted from the database will be the ones which match the entered character string.

- **Sort criteria**

Sort criteria are used to sort the data accessed from the database. A sort criterion consists of one or more fields.

To define the sort criteria for the Business Object:

1. Select *NAME* from the *Fields* list,

2. Click the corresponding >> button.

The *Sort direction* column is set to ascending by default. This is the sort order we need for this sample application. In the final application, the rows will be sorted in ascending order of the Name field.

*SQL Join Conditions* panel:

When you work with several tables, you have to define a join condition in order to specify how to synchronize data between the tables. In this sample application, the join condition will join the department number of the STAFF table and the department number of the ORG table.

To define the join condition between the STAFF and ORG tables:

1. Select the *Standard WHERE clause* radio button.
2. Select *STAFF* from the *Source Table* drop-down list, and select *DEPT* from the *Source Columns* list.
3. Select *ORG* from the *Target Table* drop-down list, and select *DEPTNUMB* from the *Target Columns* list.
4. Select the *Add join field* push-button to set the join condition.
5. Click *OK* to save all the properties specified for the Business Object and close the Definition editor.

---

## Generation Parameters

In the *Optimization* panel of the Generation Parameters editor, set the *Service level* parameter to *detail*.

Only the parts required for the detail will be generated.



---

## Chapter 17. Creating an Interface Unit Using the Sample Business Object as a Detail

To create the Interface Unit:

1. From the VAGTemplates Browser, select *New...*, from the *Instance* drop-down menu (or right click and select *New...* from the pop-up menu).  
The *New VAGT instance* window opens.
2. Enter *Detail* in the *Instance name* field, and select an application where the new Interface Unit instance will be stored in the *Application* combo box.  
For this sample application select *MdmExample1App*.
3. In the *Instance type* field, select the Interface Unit entity.
4. Select the *Open now* check box to open the Interface Unit Definition editor as the creation finishes.
5. Click *OK*.





---

## Chapter 18. Defining an Interface Unit

---

### Definition

The Interface Unit Definition editor enables you to specify the functional aspects of your Interface Unit. Use the tree view to navigate from one panel to another within the editor.

*General panel:*

1. In the *Default use name* field, type *Detail*.
2. In the *Display name* field, type *Detail*.
3. **GUI** Select the *Root interface unit* check box from the *Type* area to indicate that the Interface Unit is the entry point of the application, and that it is not called by any other Interface Unit.
4. **TUI** Before generating your TUI application, set the *Type* to *simple*. We will actually create another Interface Unit that calls this one.

*Descriptions panel:*

1. In the *Textual Description* field type *Interface Unit used to show Staff details*.
2. In the *On line help description* field:
  - if you are developing a GUI client application enter: *This window presents the details of a staff member.*
  - if you are developing a TUI application enter: *This map presents the details of a staff member.*

*Business Objects panel:*

1. To indicate which Business Object is called in the Interface Unit, click the *Add* push-button or right click in the *Business Object name* column,
2. Select *Sample* in the *Add Business Objects* window,
3. Click *OK* to save the functional properties and close the editor.

**Note:** By default, the presentation type for the Business Object is detail. This fits our example. If you want to change to a list presentation, click on the *Layout type* cell. A drop-down list appears from which you can select the list presentation.

If you added the wrong Business Object, you can remove it from the list. Select it in the list, then click on the *Remove* button, or right click, and select **Remove** from the pop-up menu.

**Note:** In our application, the Detail Interface Unit does not call any other Interface Unit. You do not have to specify any target Interface Unit.

---

## Generation Parameters

The Interface Unit Generation Parameters editor enables you to set technical and generation parameters for your Interface Unit.

*General* panel

1. In the *Title* field, type *Staff Detail*.

This title will be displayed in the window's title bar (GUI) and in the map's header (TUI) when the application is running.

2. **TUI** In the *Fastpath* field type *Detail*.

**Note:** The fastpath will enable the end user to access the map rapidly from anywhere in the application. By default the fastpath consists of the first six characters of the Interface Unit name.

3. Click *OK* to save the specified parameters and close the Generation Parameters editor.

---

## Chapter 19. TUI Only: Creating and Defining a Menu

TUI applications are often built with a main menu from which the end user can access all parts of the application. This main menu is a map that calls only maps and no Business Objects. You can also create cascading menus by defining navigation between other menus.

---

### Creating the Menu Interface Unit

To create the Menu Interface Unit:

1. From the VAGTemplates Browser, select the Interface Unit entity from the list of entities.
2. From the *Instance* drop-down menu, select *New...* (or right click and select *New...* from the pop-up menu).  
The *New VAGT instance* window opens.
3. Enter *MainMenu* in the *Instance name* field, and select an application in the *Application* combo box.  
For this sample application, select *MdmExample1App*.
4. Select the *Open now* check box to open the Interface Unit Definition editor as the creation ends.
5. Click *OK*.

---

### Defining the Menu Interface Unit

We first specify the Interface Unit functional characteristics and then its parameters.

#### Definition

Use the tree view to navigate from one panel to another within the editor.

*General* panel:

1. In the *Default use name* field type *MainMenu*.
2. In the *Display name* field type *MainMenu*.
3. Select the *Root interface unit* check box from the *Type* area to indicate that our Interface Unit is not called by any other Interface Unit.

The generated application will always start with this map, which knows all the other maps called in the application.

*Descriptions* panel:

1. In the *Textual Description* field type *Main menu for accessing the map that presents Staff details*.
2. In the *On line help description* field type *This main menu allows you to access the map that displays the details of a staff member*.

*Target Interface Units* panel:

1. To indicate which Interface Unit is called by our root Interface Unit, click the *Add* push-button or right click in the *Interface Unit* column,
2. Select *Detail* in the *Add Interface Units* window,  
If you added the wrong Interface Unit, you can remove it from the list. Select it in the list and click *Remove*, or right click and select ***Remove*** from the pop-up menu.
3. Click *OK* to save the functional properties you specified and close the editor.

**Note:** In our application, the *Menu* map does not call any Business Object. You do not have to specify any Business Object.

## **Generation Parameters**

*General* panel:

1. In the *Title* field enter *Main menu*.
2. In the *Fastpath* field type *Menu*.
3. Click *OK* to save the parameters and close the editor.

---

## Chapter 20. Generating your Application

Generating your application will transform the descriptions of instances and their relationships into operational VisualAge Generator and VisualAge Smalltalk Enterprise Parts.

---

### Generating your GUI Client Application

To generate your GUI client application, follow these steps:

1. Select the *Detail* Interface Unit from the list of Interface Unit instances,
2. In the **Instance** drop-down menu, select **Generate...**,
3. Select the *GUI InterfaceUnit - Smalltalk Oriented* generator from the list of generators.
4. Select the *cascaaded generation with predefined parts* radio-button.

All the instances you have just defined will be generated as they are directly or indirectly called by the Interface Unit.

**Note:** You only need to generate GUI predefined Parts the first time you generate a GUI application within a given Workspace, or if you change Workspace settings.

5. Click OK.

The components generated from the entity will be stored by default in the MyVAGTEntitiesApp application. You can change this default application, before generating, in the *Target Applications* panel, using the **Default Parameters** choice from the **Entity** menu for each entity (refer to the VAGTemplates Workbench Part, in your User's Guide).

The generated predefined parts will be stored by default in the MyVAGTWorkspaceEntitiesApp application. You can change this default application, before generating, in the *Target Applications* panel of the Workspace Definition editor, (**Definition** choice from the **Workspace** menu). For details, refer to the VAGTemplates Workbench Part, in your User's Guide.

For information on the available generators, refer to the Application Generation and Enhancement Chapter in the Standard Use of VAGTemplates Part, in your User's Guide.

If no error occurs during the generation, a window opens and displays the following message: "*Detail generated with InterfaceUnit - Smalltalk oriented generator*".

If errors are detected during the generation, the System Transcript displays the error messages. Correct the errors and launch the generation again.

---

## Generating your TUI Application

To generate your TUI application, follow these steps:

1. Select the *MainMenu* Interface Unit from the list of Interface Unit instances,
2. In the *Instance* drop-down menu, select **Generate...**,
3. Select the *TUI Interface Unit Logic* generator from the list of generators.
4. Select the *cascaded generation with predefined parts* radio-button.

All the instances you have just defined will be generated from the Interface Unit as they are directly or indirectly called by the Interface Unit.

**Note:** You only generate TUI predefined Parts the first time you generate a TUI application within a given Workspace, or if you change Workspace settings.

For information on the various available generators, refer to the Standard Use of VAGTemplates Part, in your User's Guide.

5. Click *OK*.

The components generated from the entity will be stored by default in the MyVAGTEntitiesApp application. You can change this default application, before generating, in the *Target Applications* panel using the **Default Parameters** choice from the *Entity* menu (refer to the VAGTemplates Workbench Part, in your User's Guide).

The generated predefined parts will be stored by default in the MyVAGTWorkspaceEntitiesApp application. You can change this default application, before generating, in the *Target Applications* panel in the Workspace Definition editor (**Definition** choice from the *Workspace* menu). For details, refer to the VAGTemplates Workbench Part, in your User's Guide.

If no error occurs during the generation, a window opens and displays the following message: "*MainMenu generated with Interface Unit Logic*".

If errors are detected during the generation, the System Transcript displays the error messages. Correct the errors and launch the generation again.

For information on the available generators, refer to the Standard Use of VAGTemplates Part, in your User's Guide.

---

## Chapter 21. Testing your Application

Once your application is generated and exported you can test it with the VisualAge workstation.

Before testing your application, you must set database preferences.

1. Select the *VAGEN Preferences* choice from the *Options* menu in the VisualAge Organizer
2. In the *VisualAge Generator Preferences* window, enter the name of the database you will access and your user id and password for the database connection in the *SQL* panel.

---

### Testing the GUI Client Application

To test your application, follow these steps:

1. In the VisualAge Organizer, select the *MyVAGTEntitiesApp* application from the *Applications* list.
2. Select the *DetailInterfaceUnitView* part from the *Parts in 'MyVAGTEntitiesApp'* list.
3. Click with left mouse button and select the *Test* choice from the pop-up menu, or click the test push-button from the VisualAge Organizer tool bar to start the test.

The Staff Detail window opens on an empty detail.

Only the fields having default values are filled.

For more information on the default value mechanism, refer to the VAGTemplates Workbench Part, in your User's Guide.

Look at the menu bar. It has: a *Sample* drop-down menu to perform actions on the data; an *Edit* drop-down menu to copy, cut and paste the contents of the input areas; a *Windows* drop-down menu to navigate towards windows that might be called by our Staff Detail window; and a *Help* drop-down menu to get help on the window if the help has been generated.

For information on generated applications, refer to the Standard Use of VAGTemplates Part in your User's Guide.

#### *Consulting Staff Member Data*

To *consult* a staff member's data:

1. Type *180* in the *Id Number* field.
2. Select *Read* from the *Sample* drop-down menu.
3. The Staff detail window displays the data corresponding to the staff member number 180.

### *Updating an Existing Staff Member*

To *update* an existing staff member:

1. Type *180* in the *Id Number* field.
2. Select *Read* from the *Sample* drop-down menu.
3. Update the values in the updatable fields; for example, enter 8 in the *Years* field.
4. Select *Update* from the *Sample* drop-down menu.

### *Creating a New Staff Member*

To *create* a new staff member:

1. Select *New* from the *Sample* drop-down menu.  
An empty detail opens.
2. Complete the detail.
3. Select *Create* from the *Sample* drop-down menu to save the data for the new staff member.

### *Deleting a Staff Member*

To *delete* a staff member:

1. Type *360* in the *Id* field.
2. Select *Delete* from the *Sample* drop-down menu.

### *Testing the Input Check*

Here, you will test three types of errors. Once you have tested one error, do not correct it so that the error window contains several error messages.

To test a **format check**:

1. In the *Id* field, type *AAA* and press the tab key to give the focus to another field.

The *Id* field background color changes and displays **\*\* error \*\*** to show that you made an input format error. In this case, you entered an alphabetical character in a field that only accepts numeric values. However, you did not specify any check for the Data Element. VisualAge Generator behaves like this due to internal controls.



You will obtain the same behavior if you type *AAA* in any numeric field (Department Number, Years or Salary).

2. Select **Check** from the **Sample** drop-down menu.

The window showing the list of input errors appears. Each item in the list indicates which field is erroneous (here, Id Number) and the nature of the error. In the case of a format error the message is:

" Id Number: value format is not valid ".

To test a **check by value table**:

1. In the *Job Title* field, type *Mrkt* and press the tab key to give the focus to another field.

The *Job title* field background color changes to show that you made an input error. In this case, you entered a non-authorized value. Remember, for the Job1 Data Element, you specified a check by value table. This check has detected that *Mrkt* is not an authorized value.

If you click on the drop-down list button you can see the list of authorized values. Select a job in the list to assign it to the staff member.

2. Select **Check** from the **Sample** drop-down menu.

The window showing the list of input errors appears. Each item in the list indicates which field is erroneous (here, Job Title) and the nature of the error. In this case the message is:

" Job Title: value is not in value table ".

To test a **check by interval**:

1. In the *Years* field, type *120* and press the tab key to give the focus to another field.

The *Years* field background color changes to show that you made an input error. In this case, you entered a non-authorized value. Remember: for the Years Data Element, you specified an interval check. This check has detected that 120 was greater than 100; the maximum value of the interval.

2. Select **Check** from the **Sample** drop-down menu.

The window showing the list of input errors appears. Each item in the list indicates which field is erroneous (here, Years) and the nature of the error. In this case the message is:

" Years: value must be between 1.0 and 100.00 ".

**Note:** When you double-click on a line of error list window, the cursor is automatically positioned in the corresponding erroneous field.

---

## Testing the TUI Application

To test your application, follow these steps:

1. In the VisualAge Organizer, select the *MyVAGTEntitiesApp* application from the *Applications* list.
2. Select the Program part from the *Parts in 'MyVAGTEntitiesApp'* list.
3. Right-click on the MAINMAM Program in the *VAGenParts* list and select the *Test* choice from the pop-up menu, or click the test push-button in the Program editor, to start the test.
4. In the *Test Monitor* window that opens, click the *Run* push-button to start the test.

The main menu opens.

In the header part of the map you can see:

- the type of the system on which the application is running,
- the user identification,
- the Workspace name,
- the map's title,
- the map's fastpath,
- the current date and time.

In the trailer part of the map, you can see the function keys and their labels that have been generated by default for your application. The *Help* key allows the end user to get help on the map if the help has been generated.

**Note:** These keys and labels can be parameterized in the VAGTemplates Workbench. For information, refer to the VAGTemplatesWorkbench Part, in your User's Guide

In the body of the map, you can see:

- the fastpath and the name of the called map. The end user can type the selection code, *1*, in the *Select* area to move directly to the Staff Detail map; he/she can also type the fastpath in the *Fastpath* area.
- the logical key of the Business Object. This logical key is laid out as an access parameter as it can be used to directly access the data from a member of staff in the Staff Detail map. The end user only needs to type the value of a key in the underscored area next to the *Fastpath* area.

For information on generated applications, refer to the Standard Use of VAGTemplates Part, in your User's Guide.

### *Opening the Staff Details Map*

There are several ways to open the Staff Details map from the Menu map:

1. Type *D* in the *Select* area and press enter.

The Staff Detail map opens on an empty detail.

2. Type *detail* in the *Fastpath* area and press enter.  
The Staff details map opens on an empty detail.
3. Type *detail* in the *Fastpath* area and type the Id number *180* in the underscored area next to the *Fastpath* area (called the *access parameters* area), and press enter.  
The Staff details map opens and displays the data from the staff member number 180.

### ***Consulting Staff Member Data***

To *consult* staff member data:

1. Type *180* in the *Id Number* field.
2. Press enter.
3. The Staff Detail map displays the complete information on staff member 180.

### ***Updating an Existing Staff Member***

To update an existing member of staff:

1. Type *180* in the *Id Number* field.
2. Press enter.
3. Update the values of the updatable fields; for example, enter *8* in the *Years* field.
4. Type the *U* action code in the *Action* area.

### ***Creating a New Staff Member***

To create a new staff member:

1. Type the *I* action code in the *Action* area.  
You can complete the new detail by overriding the data in the detail.
2. Type *360* in the *Id Number* field.
3. Press enter.

### ***Deleting a Staff Member***

To delete a member of staff:

1. Type *360* in the *Id Number* field.
2. Press enter.
3. Type the *D* action code in the *Action* area.

### ***Testing the Input Check***

Here, you will test three types of errors. Once you tested one error, do not correct it so that the error map contains several error messages.

- To test a **format check**:

Try to type a non-numeric character in the *Id Number* field.

The control will not let you since this field, as a numeric field, does not allow other character input. You will obtain the same behavior with any numeric field (Department Number, Years or Salary).

- To test a **check by value table**:

Type *Mrkt* in the *Job Title* field, and press enter.

The *Job Title* field color changes to show that you have made an input error. In this case, you entered a non-authorized value. Remember: for the Job1 Data Element, you specified a check by value table. This check has detected that *Mrkt* is not an authorized value.

The map showing the list of input errors appears. Each item in the list indicates which field is erroneous (here, *Job Title*) and the nature of the error. In this case the message is:

" Job Title: not in value table ".

- To test a **check by interval**:

In the *Years* field, type *120* and press enter.

The *Years* field color changes to show that you made an input error. In this case, you entered a non-authorized value. Remember: for the Years Data Element, you specified an interval check. This check has detected that 120 was greater than 100, the maximum value of the interval.

The map showing the list of input errors appears. Each item in the list indicates which field is erroneous (here, *Years*) and the nature of the error. In the case of a format error the message is:

" Years: value must be between 1.0 and 100.00 ".

### *Testing the Input Aid*

Remember, we have defined the Job1 Data Element with a check by value table and we have filled the value table with several values. In the final application, the end user is aided with the list of authorized values.

To view the list of authorized values:

1. Put the cursor in the *Job Title* field,
2. Press the F04 function key, which is laid out as *LOOKUP*.

A new map is displayed around the *Job Title* field and shows the list of authorized values.

```

SYST: DS2CICS          MYAPPLICATION          FASTPATH: Detail
USER: PC user          Staff details          05-27-1997 12:21:12

ACTIONS: I INSERT U UPDATE D DELETE
ACTION: _
ID NUMBER             180
NAME
DEPARTMENT NU
JOB TITLE
YEARS
SALARY
DEPTNUMB
LOCATION

FASTPATH: _____

VALUE TABLE PANEL
  Mgr
  Clerk
  Sales
  Mkt
  _
  _
  _
  _
  _
  _
  PF03: EXIT  PF07:  PF08:

PF01: HELP  PF02:  PF03: EXIT  PF04: LOOKUP  PF05:  PF06:
PF07:  PF08:  PF09:  PF10:  PF11:  PF12: CANCEL

```

3. Select one of the values by typing any character in front of it and press enter.

The map closes and the field is updated.

**Note:** You can type any character in the action area of the value table panel to select a value, but if you leave it empty, no value is selected and the panel closes.

To close the value table map and go back to the previous map, press the F03 function key.



---

## Chapter 22. Exploring the Generated Parts

You will need to know the generated components when you begin to develop your own applications to determine which components are customizable. For this sample application, we will not present them exhaustively here. You can find the complete information in the Standard Use of VAGTemplates Part, in your User's Guide.

**Note:**

The Parts generated from the Interface Unit, Data Element, Business Object, Relational Table, and Value Style entities are stored in the default MyVAGTEntitiesApp application.

The predefined parts, generated from the Workspace entity, are stored in the default MyWorkspacePartsApp application.

---

### Exploring the Parts Generated by the GUI Generators

#### *Generated Data Items*

A Data-Item is generated for each Data Element used in the application.

*DEPTDE* is the name of the Data Item generated from the *DEPT* Data Element.

DEPT is the Data Element name

D is the Data Item generation mnemonic

E is the Data Element generation mnemonic

#### *Generated Visual Parts*

The following parts are visual parts:

- *SamplDetailSubview*, the Visual Part generated for the detail subview.
- *SamplListSubview*, the Visual Part generated for the list subview.
- *SamplUpdatableListSubview*, the Visual Part generated for the updatable list subview.

**Note:** The detail, list and updatable list parts are not always generated. Their generation depends on the value set for the *service level* parameter (see the Part dedicated to the VAGTemplates Workbench, in your User's Guide).

- MyAppErrorView and MyAppSystemErrorView, the Visual Parts generated for the error views.
- DetailInterfaceUnitView, the Visual Part generated for the Detail Interface Unit.

Access the definition of this part by double-clicking on it in the *VAGen Parts* list, in the VisualAge Organizer.

The DetailInterfaceUnitView Part contains:

- the description of the SampleDetailSubview Part. You access it by double-clicking in the form embedded in the DetailInterfaceUnitView Part.
- the variable based on the Business Object Part, which represents the Sample Business Object.
- Select this variable and right click. The *Start connection from* window opens displaying the list of available actions, attributes and events defined for the Business Object.

### *Generated Classes*

The following classes are non visual parts:

- SampleBusinessObject, the Business Object Part, which encapsulates the Business Object's data and the actions that manage this data.
- SampleListManager, the Part that manages multi-instance Business Objects and the associated paging actions.
- SampleUpdatableListManager, the Part that manages updatable multi-instance Business Objects and the associated update and paging actions.
- SampleResourceObject, the Part that encapsulates mono-instance and multi-instance server accesses.
- MyAppErrorHandling, the Part that implements the Data Elements checks and corrects the raised errors.

### *Generated Programs*

The following are the VisualAge Generator Programs:

- SAMPLA1, the Program (server application) generated for the mono-instance Business Object.  
This Program manages data access in a detail Business Object. It includes a number of hooks in which you can enter specific code.
- SAMPLAN, the Program (server application) generated for the multi-instance Business Object.  
This Program manages data access in a Business Object list. It includes a number of hooks in which you can enter specific code.



- *SAMPLAL*, the Program (server application) generated for the updatable multi-instance Business Object.

This Program manages data access in an updatable list Business Object. It includes a number of hooks in which you can enter specific code and it commits updates of the updatable list.

---

## Exploring the Parts Generated by the TUI Generators

### *Generated Programs*

The generated Programs are the following:

- *MAINMAM*, the Program generated for the menu map.
- *DETAIAM*, the Program generated for the Staff Detail map.
- *SAMPLA1*, the Program (server application) generated for the mono-instance Business Object.

This Program manages data access in a detail Business Object. It includes a number of hooks in which you can enter specific code.

- *SAMPLAN*, the Program (server application) generated for the multi-instance Business Object.

This Program manages data access in a Business Object list. It includes a number of hooks in which you can enter specific code.

- *SAMPLAL*, the Program (server application) generated for the updatable multi-instance Business Object.

This Program manages data access in an updatable list Business Object. It includes a number of hooks in which you can enter specific code.

- *SAMPLAF*, the Program generated for the Data Elements called in the Business Object.

This Program manages data control in an updatable list Business Object. It includes a number of hooks in which you can enter specific code.

- *SSYSERT*, the Program generated for the system error map.

### *Generated Maps*

The generated maps are the layouts of the final application interface.

The following Map Parts are generated for the MainMenu Interface Unit:

- *MAINMB HEADER* is the header layout (1). It displays information such as the system on which the application is running, the user identifier, the current map fastpath, and the actions available on the map.
- *MAINMB MAP1* is the layout of the map body (2). It displays the list of child maps and the list of access parameters for the Business Object data, the selection area, and the fastpath and access parameters input area.

- MAINMB TRAILER is the trailer layout (3). It displays the available functions keys defined for the map and includes a blank line where the error messages are displayed if you specify it.

```
#SYST:^          #          ^          #          #FASTPATH:^      #
#USER:^          #          ^          #          #          # (1)

#S#FASTPATH#          #ACCESS PARAMETERS          #
^ ^          ^          ^
#^ ^          ^          ^
#^ ^          ^          ^
#^ ^          ^          ^
#^ ^          ^          ^
#^ ^          ^          ^
#^ ^          ^          ^
#^ ^          ^          ^
#
#SELECT:^ #

#FASTPATH:^          #          # (2)
^ ^
#PF01:^          #PF02:^          #PF03:^          #PF04:^          #PF05:^          #PF06:^          #
#PF07:^          #PF08:^          #PF09:^          #PF10:^          #PF11:^          #PF12:^          # (3)
```

These three Map Parts make up the Main Menu map that the end user will use to access all other maps in the final application.

- MAINMB SERRLST is the layout of the pop-up error map. It displays the map title, and the available functions keys defined for the map. It includes blank lines where the error messages will be displayed.

```
^          # ERRORS LIST: ^
^ ^
^ ^
^ ^
^ ^
^ #
^ #PF03:^          #PF07:^          #PF08:^          ^          #
^          #
```

- S SSYSER is the layout of the pop-up system error map. It displays the map title, the returned error code(s) and the name of the application where the error occurred and the Function that was running.

```
#SYSTEM ERROR INFORMATION PANEL#

#SQL CODE:^          #          ^          #
#APPLICATION NAME:# ^          #          #
          #FUNCTION NAME....:# ^          #

          #(Press any key to Exit)          #
```

The following Map Parts are generated for the Detail Interface Unit:

- DETAIB HEADER is the header layout (1). It displays information such as the system on which the applications is running, the user identifier, the current map fastpath, and the actions available on the map.
- DETAIB MAP1 is the layout of the map body (2). It displays the Business Object fields, the action area, and the fastpath and access parameters input area.
- DETAIB TRAILER is the trailer layout (3). It displays the available function keys defined for the map and includes a blank line where the error messages are displayed if you specify it.

```

#SYST:^          #          ^          #          #FASTPATH:^          #
#USER:^          #          ^          #          ^          #          #
#ACTIONS:^ ^    ^ ^    ^ ^    ^ ^    ^ ^    #          (1)
#ACTION:^ #
#ID NUMBER^    #
#NAME#        ^          #
#DEPARTMENT NUMBER^    #
#JOB TITLE#    ^          #
#YEARS#        ^          #
#SALARY#       ^          #
#DEPTNUMB#     ^          #
#LOCATION#      ^          #

#FASTPATH:^    # ^          #          (2)
^          ^          #          #
#PF01:^        #PF02:^    #PF03:^    #PF04:^    #PF05:^    #PF06:^    #
#PF07:^        #PF08:^    #PF09:^    #PF10:^    #PF11:^    #PF12:^    # (3)

```

These three Map Parts make up the Staff Detail map that the end user will use to access staff member data.

- DETAIB SERRLIST is the layout of the pop-up error map. It displays the map title, and the available function keys defined for the map. It includes blank lines where the error messages will be displayed.

```

^          # ERRORS LIST: ^
^ ^
^ ^
^ ^
^ ^
^ #
^ #PF03:^    #PF07:^    #PF08:^    ^
^          #          #          #
#

```

- DETAIB SHELP is the layout of the pop-up on-line help, help lists. It displays the available functions keys defined for the map and includes blank lines where the help text will be displayed.



---

## Part 3. Appendixes



## Parameterizing DB2 Database Import

DB2 enables the parameterization of the system tables defined by the import to the name of the access views, thanks to VAGTSys variable.

- Use of SYSSCHEMA keyword (DB2CLI.INI in your DB2 directory)

This keyword indicates an alternative schema to be searched in place of the SYSIBM (or SYSTEM, QSYS2) schemas when the DB2 CLI and ODBC Catalog Function calls are issued to obtain system catalog information.

Using this schema name, the system administrator can define a set of views consisting of a subset of the rows for each of the following system catalog tables:

DB2 for common server	DB2 for MVS/ESA	DB2 for VSE and VM	DB2 for OS/400
SYSTABLES	SYSTABLES	SYSCATALOG	QADBXFIL
SYSCOLUMNS	SYSCOLUMNS	SYSCOLUMNS	QADBIATR
SYSINDEXES	SYSINDEXES	SYSINDEXES	QADBKATR
SYSTABAUTH	SYSTABAUTH	SYSTABAUTH	QADBRKCL
SYSRELS	SYSRELS	SYSKEYCOLS	QADCKCL
SYSDATATYPES	SYSSYNONYMS	SYSSYNONYMS	QADBLDEP
	SYSKEYS	SYSKEYS	
	SYSCOLAUTH	SYSCOLAUTH	
	SYSFOREIGNKEYS		
	SYS PROCEDURES (1)		
	SYS DATABASE		

**NOTE:** DB2 for MVS/ESA 4.1, only.

For example, if the set of views for the system catalog tables are in the ACME schema, then the view for the SYSIBM.SYSTABLES is ACME.SYSTABLES, and SYSSCHEMA should then be set to ACME.

Defining and using limited views of the system catalog tables reduces the number of tables listed by the application, which reduces the time it takes for the application to query table information.

If no value is specified, the default is:

- SYSCAT or SYSIBM on version 2.1 of DB2 for common server

- SYSIBM on versions prior to 2.1 of DB2 for common server, DB2 for MVS/ESA and OS/400
- SYSTEM on DB2 for VSE and VM
- QSYS for OS/400.
- Use of VAGTSys variable (DB2 for MVS/ESA only)  
DB2 for MVS/ESA allows you to create views for system tables with a parameterized table name (e.g. DSNCOLUMNS in place of SYSCOLUMNS).  
The VAGT Relational Import function is able to use such parameterized names: you just have to initialize VAGTSys variable in the DB2CLI.INI file, according to your parameterization (e.g. VAGTSys = DSN).  
If no value is specified, the default is: SYS.



---

# Readers' Comments — We'd Like to Hear from You

VisualAge Generator Templates  
Introducing VisualAge Generator Templates  
Version 4.5

Publication No. GH23-0272-01

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Complete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to find	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to understand	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Well organized	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Applicable to your tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Please tell us how we can improve this book:

Thank you for your responses. May we contact you?  Yes  No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

---

Name

---

Address

---

Company or Organization

---

Phone No.



Fold and Tape

Please do not staple

Fold and Tape



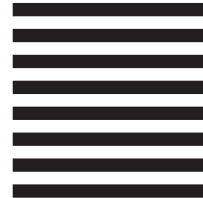
NO POSTAGE  
NECESSARY  
IF MAILED IN THE  
UNITED STATES

# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation  
IBM SWS - Paris Laboratory  
1, place Jean-Baptiste Clément  
93881 - Noisy-le-Grand, France



Fold and Tape

Please do not staple

Fold and Tape





Part Number: CT7P8NA



Printed in the United States of America  
on recycled paper containing 10%  
recovered post-consumer fiber.

GH23-0272-01



(1P) P/N: CT7P8NA

