Composer provides a comprehensive set of engineering and maintenance tools for the Symphony Enterprise Management and Control System. Composer is designed to operate on the Microsoft® Windows 2000 (Service Pack 2) or Windows NT® 4.0 (Service Pack 6A or later) platform.

The working environment provided by Composer simplifies the configuration and maintenance of Symphony systems. The base product contains all the functionality necessary to create and maintain control system configurations. Applications provide users with the ability to graphically develop control system strategies, develop and maintain global configuration databases, and manage system libraries of reusable software components.

Composer is designed to be compatible with INFI 90 OPEN system configurations and is capable of importing existing WinTools configurations. Once imported, these configurations can be fully integrated into Composer and utilize all its features.
Additional Features

- **Multiuser, client/server architecture**: Composer applications use client/server technology to support multiple users operating in a networked environment. Configuration information which is managed by Composer’s configuration server can be accessed simultaneously by multiple users. In addition to interacting with configuration information, users can access online data from a running Symphony or INFI 90 OPEN system by using Composer’s communication server. The Composer architecture supports one configuration server per system and multiple communication servers.

- **Configuration database**: A configuration server can support up to ten simultaneous client connections and provide users with shared access to a system’s configuration information. Composer’s configuration server manages and stores configuration data in one configuration database per project or system. This configuration database eliminates duplication of data entry, simplifies database management, and automates many configuration tasks. Information can be imported and exported in many formats.

- **Shared network interface devices**: Communication servers provide users with the ability to communicate with a running Symphony or INFI 90 OPEN system by accessing the Control Network (Cnet) via the Operations Network (Onet). Network couplers or computer interface devices can be simultaneously shared by Composer clients via the Onet.

- **Object exchange**: Composer introduces a single, system-wide component database which is called the object exchange. Object exchange provides a multiuser repository for all standard symbols, macros, control strategies and control logic templates that are used to generate control system strategies. The object exchange is a prominent part of Composer client applications. Users are encouraged to use system and project standards when creating automation strategies. In addition to presenting standard system objects, the object exchange enables users to add components they have created in the object exchange.

- **Custom C programs**: Load custom C programs to the controllers. Composer’s base client provides the ability to load custom C programs without the need for an additional utility. Examples of custom applications include foreign device interfaces and performance calculations.

- **Batch data manager**: Batch data manager (BDM) is a family of engineering tools for creating, editing, managing, downloading and debugging batch, sequential, and user defined function (UDF) code configurations. BDM supersedes and provides migration from all previous batch and UDF tools. BDM is comprised of three product options: Batch 90 for phase execution, batch tools for batch sequencing, and UDF codes which are discussed in more detail in the *Optional Client Applications* section of this overview.

Composer Applications

The base Composer product contains all the functionality necessary to develop and maintain Symphony control system configurations. There are two primary applications: explorer and automation architect.

Exploring

The primary application of Composer is the explorer. Explorer presents the Symphony system architecture and provides an intuitive means for organizing, navigating and locating system
configuration information. Explorer presents a user with two main windows: system architecture and the object exchange.

**System Window**

The system architecture window functions similarly to Microsoft’s file explorer. The left pane of the window displays a hierarchical representation of the Symphony system. When a system object is selected, the right pane displays a detailed view for the selected object.

The system window supports two views: the document view (Fig. 1) and the data browser view (Fig. 2). When the system window is in the document view, it will show the configuration documents that are associated with the system object that the user has selected. Configuration documents support long file names and can include control logic documents, human system interface displays, or documents created by other applications such as CAD packages or spreadsheets.

![System Architecture - Document View](image)

**Figure 1. System Architecture - Document View**

The ability to associate any documents with the system architecture is an important feature. This allows any information, such as P&IDs, cabinet arrangement drawings or field wiring drawings, to be managed by the configuration server and therefore accessed by Composer client applications. All that is required to edit any of these documents is to double click on the document. Composer’s explorer will automatically launch the appropriate application for the document selected.

When the system window is in the data browser view, the right pane of the system architecture window will display tag information associated with the system object the user has selected. All tag information presented is retrieved from the configuration server database that is managed by the Composer server.

When working in the data browser view, users can view, define, and modify tag data for the whole system. This central repository of data is managed by Composer’s configuration server for all tag data in the entire system. The data for each tag is added to the configuration server database as each tag is defined. This eliminates the need for users to enter the same information more than once. Some notable features of the data browser view are the ability to:
Edit tag objects in a datasheet or property page view.

Filter the database. Filtering makes configuration easier and faster by eliminating unnecessary information from view.

Import and export tag data.

Navigate directly from a tag to its related configuration document.

Perform automatic search and replace operations based on complex queries.

Object Exchange

The object exchange (object library) window presents the user with a view of the reusable components that can be used to create control system configurations (Fig. 3). Objects are organized in folders. Standard system components such as function codes and standard shapes and symbols are organized under the system folder. Users are able to use these components, but they are not permitted to delete them from the object exchange because they are part of the standard system objects supported by Composer.

Users can define their own folders within object exchange. User-defined folders can be nested as deeply as the user desires and provide a container for user-defined objects. User-defined objects can include macro logic, shapes, symbols, standard drawings or control logic templates. In addition, user-defined folders can contain references or shortcuts to system defined objects. This enables users to effectively organize standard system objects to best suit their needs.

To support reuse, the object exchange provides library management features such as cutting, copying, and pasting of objects between different projects. This makes it easy for systems engineers to share objects among projects.
Automation Architect

The automation architect provides for the visual creation, editing, monitoring, and tuning of control logic. High-level control strategies can be created by dragging and dropping function codes from the object exchange to the control logic document.

Control strategies are represented graphically by the automation architect. Rather than textually programming strategies, the automation architect represents predefined control strategies as function blocks. By connecting function blocks (Fig. 4), users are able to specify the signal flow of a control strategy and visually define the control strategy.
The automation architect stores configuration information in control logic documents. Control logic documents support grouping of multiple logic sheets in a single document. This permits users to group sheets of logic together using process partitions. For example, a single control logic document could be used to define the control strategy for a mix tank. Each control loop or motor control sheet associated with the mix tank could be assigned to the control logic document. Partitioning control logic in this manner is more process object oriented and intuitive to process engineering personnel (Fig. 5).

![Figure 5. Control Logic Document Sheets](image1)

The monitoring and tuning capabilities (Fig. 6) of the automation architect provide the ability to troubleshoot and maintain an operational system using the same information used to create the system. By using the monitoring functionality, it is possible to obtain dynamic operating values from the Symphony system. These values are automatically presented on the same control logic documents that were used to configure the module. Composer’s tuning functionality allows the change of logic parameters as permitted by the controller. The control logic document in the Composer application and in the module are dynamically updated when tuning changes are made so that the documentation for the system accurately reflects the current configuration of the controller.

![Figure 6. Monitoring and Tuning Capabilities](image2)

**Control Logic Templates**

Two of the primary goals of Composer are to reduce the cost of implementing control strategies and to improve the quality of Control strategy software. To realize these goals, Composer supports a new type of document called a Control Logic Template.

Control Logic Templates (Fig. 7) define reusable standard control strategies that are typically used to develop a process automation system. They can be thought of as blueprints that define the
structure of a control strategy. They are maintained by the object exchange and can be used to quickly define control logic documents.

The Control Logic Template Linking functionality allows users to define logic that is controlled by the template or can be modified within a logic document. Any subsequent changes made to a template can then be propagated to logic documents. When a template updates its documents, it will preserve logic additions that the user has made to the document. This template management functionality provides efficient maintenance and utilization of reusable standard control logic.

Figure 7. Control Logic Templates

Optional Client Applications

HSI Options

Graphic configuration applications are options that can be added to the base Composer product. These applications create interactive operating views of the process being controlled using standard and custom symbols and shapes. These views are used to operate processes controlled by the Symphony system. The graphical display configuration (GDC) application is used when creating displays for the Conductor VMSTM or INFI 90 OPEN human system interfaces. The graphic con-
configuration application (GrafX) is required when creating Operate IT™ or Conductor NT displays (Fig. 8).

![Graphic Configuration Application (GrafX)](TC00994A)

**Figure 8. Graphic Configuration Application (GrafX)**

### OLE Interface

All of the controller configuration and tag configuration information is available via the OLE interface client option of Composer. Using the OLE interface, users can access, extract and replace configuration information stored in the Composer configuration server. Any application that is capable of communicating via OLE/COM can access information from the Composer configuration server via the OLE interface. Examples of applications that are capable of accessing information via OLE include Visual Basic™, Visual Basic for Applications in Microsoft Office, or web pages using VBScript or JScript®.

The primary advantage of accessing data via Composer’s OLE interface is allowing users to extend the capabilities of Composer with custom scripts or add-on applications. At the same time, users are able to rely upon the extensive data concordance rules provided by the Composer configuration server.

### Batch Data Manager

Batch data manager (BDM) is a family of engineering tools for creating, editing, managing, downloading and debugging batch, sequential and user defined function code configurations.

The heart of BDM is the Batch 90 programming language. Batch 90 enables the engineer to create batch and sequential control applications using clear, concise, natural syntax control statements. Batch 90 is layered on top of regulatory and discrete device control function code logic. Batch 90 programs can be used to change controller set points, turn discrete devices on and off, change modes, and perform a host of other supervisory operations. In effect, function codes execute base
regulatory and discrete device control while the Batch 90 program performs supervisory control and process operation.

A Batch 90 program is comprised of all the phases that can be run on a specific class of equipment. A phase is the smallest element of control that can accomplish a process-oriented task on a specific class of equipment.

A Batch 90 phase provides a structured, modular approach to batch design by including distinct logic sections for:

- Normal logic executes standard sequential control.
- Continuous control logic continuously executes a set of interlocks.
- Fault logic executes if a stop command is issued or if a fault is detected in normal logic or continuous control logic.
- Hold logic executes if a hold command is issued.
- Restart logic executes return-to-normal logic from the held or stopped states.

Batch 90 supports parallel execution of up to 32 phases, unit relative programming that allows Batch 90 programs to be written for classes of equipment, abort logic specific to each class of equipment, and synchronization between batch programs running in different areas of a plant.

Recipes are used to determine the procedure and formulation of the batch. A procedure determines the order in which the Batch 90 phases are executed. Formulation is the set of data that defines the requirements for a specific product. For example, formulation could include target flow rates, quantities, temperatures and times.

BDM applications reference their configurations in projects. This is similar to the Composer concept of a project, but BDM projects are managed separately from Composer projects. A BDM project can be located on the computer local to the user, or it can be accessed remotely over a network to a mapped drive if a file server is used. A project typically encompasses a process area. A UDF project includes the required UDF programs for a process area. A batch project references a class library that includes a class for each type of equipment in a process area. Each class contains the Batch 90 program associated with the class of equipment plus the unit definition files if unit relative programming is implemented. If the unit procedure editor or master recipe editor is used, recipes are created and managed in a separate window of the same project.

BDM is comprised of three product options:

- **Batch 90 for Phase Execution.** Batch 90 for phase execution includes a full-featured text editor to create and compile Batch 90 programs that are downloaded to the phase execution (PHASEX) function code (Fig. 9). In this batch execution model, recipes are created and run using ABB batch management software. The batch manager sends the phase name and formulation data of the active phase in the recipe to the PHASEX function block and issues a start command. The PHASEX block executes the phase logic. The batch manager monitors the state of the PHASEX block and then automatically transitions to the next phase defined in the recipe when the complete state is detected for the phase. This execution model provides flexible configuration and easy modification of recipes. It is especially useful where many different products are produced, frequent product formulation changes occur, or different units can be used to make a specific product.

- **Batch Tools for Batch Sequencing.** Batch tools for batch sequencing include a full-featured text editor to create and compile Batch 90 programs that are downloaded to the batch
sequence (BSEQ) function code. Also included are the unit procedure editor (Fig. 10) and the master recipe editor which are respectively used to create unit procedures and master recipes in a graphical format. Creation of master recipes is optional; they provide a method of linking unit procedures for a specific production train. In this batch execution model, recipes are created using BDM and downloaded to the controller in their entirety before the batch begins. This execution model provides high system integrity since all the information needed to execute the batch is self-contained in the controller. It is especially useful where a limited number of products are produced or for sequence control applications where the sequence is rarely modified.

- **User Defined Function Codes.** User defined function (UDF) codes includes a full-featured text editor to create and compile UDF programs using natural syntax control statements similar to Batch 90. UDF programs can be downloaded to UDF-specific function codes (Fig. 11) and incorporated into control strategies in the same manner as any other function code. The UDF program does not require a recipe or an operator input; it begins executing when the controller begins executing. This product is especially useful for users who need to create function codes unique to their process or for sequence control applications that execute continuously and do not require a recipe.

The dynamic debugger, the primary troubleshooting tool for Batch 90 and UDF programs, is used with all three product options. The dynamic debugger (Fig. 12) enables engineers to:

- Observe process data such as variables and set points, as well as intermediate computational values from the program including timers, ramps and integrators.

- Override normal operation of pumps, valves and control loops, and direct their activities from the debugger.

- Stop normal program operation and single-step the program to analyze data without affecting other programs or function block execution.

- Assign breakpoints. The program can be designed to run to a certain point and then stop execution, allowing engineers to systematically analyze logic.
Perform online recipe parameter editing when used with the unit procedure editor or the master recipe editor.
System Architectures

Composer can be applied to existing system architectures or new system architectures. Figure 13 shows a possible application of Composer in an existing system to replace an INFI 90 OPEN engineering workstation. In this configuration the Composer client and server applications can be loaded onto the same physical machine. Configuration files that are managed by the configuration server are resident on the hard drive of the composer machine. Access to the local ICI is achieved via the Composer communication server that is loaded and running on the Composer communication server. Client applications provide users with the ability to view and edit configuration information from this machine.

The Composer architecture is expandable to meet users’ changing needs. If a user wanted to expand the architecture to accommodate additional users, it could be expanded as shown in Figure 14. In this setting the user has purchased two additional clients and expanded the server license to accommodate three client connections. In addition, the user has purchased an additional ICI coupler, networked the machines via TCP/IP on Ethernet™, and added a new Symphony PCU. In this architecture, any of the client machines must be able to access the configuration server to display configuration information. All clients can access the configuration server simultaneously. Clients can select the communication server that they wish to use to interact with the run-time server. Even client machines which do not have a local connection to the ICI coupler can access the run-time system by connecting the ICI coupler of another machine via the Ethernet.

International Language Support

The base Composer applications have been developed to support international languages. Configuration documents developed using Composer applications will accept and preserve user inputs in any language supported by the Windows 2000 or the Windows NT operating system (Fig. 15). When a user creates a project, Composer’s configuration server sets the default language for the project to match the default language of the user creating the project. Once the project is created,
users can enter text into the documents and database records presented by Composer client applications. While Composer supports any of the languages supported by Windows 2000 or Windows NT, users that require international language support should consult their regional ABB office to ensure that the specific products they want to configure supports a specific language.

**Online Documentation**

Composer provides users with standard Windows help. In addition to this, all Composer documentation is provided in electronic form (Fig. 16). The instruction manuals for Composer are provided on the product CD disk in Adobe® PDF format along with an enhanced version of Adobe Acrobat® reader that supports a sophisticated search engine.
Figure 14. System Architecture - Multiple Users
Figure 15. International Language Support

Figure 16. Online Documentation