

ANSI/ISA-88.00.02-2001



Batch Control Part 2: Data Structures and Guidelines for Languages

NOTICE OF COPYRIGHT

This is a copyrighted document and may not be copied or distributed in any form or manner without the permission of ISA. This copy of the document was made for the sole use of the person to whom ISA provided it and is subject to the restrictions stated in ISA's license to that person. It may not be provided to any other person in print, electronic, or any other form. Violations of ISA's copyright will be prosecuted to the fullest extent of the law and may result in substantial civil and criminal penalties.



ISA—The Instrumentation,
Systems, and
Automation Society

Approved 7 February 2001

ANSI/ISA-88.00.02-2001

Batch Control Part 2: Data Structures and Guidelines for Languages

Copyright © 2001 by ISA-The Instrumentation, Systems, and Automation Society. All rights reserved. Not for resale. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), without the prior written permission of the Publisher.

ISBN: 1-55617-745-3

ISA
67 Alexander Drive
PO Box 12277
Research Triangle Park, NC 27709
USA

Preface

This preface, as well as all footnotes and annexes, is included for information purposes and is not part of ANSI/ISA-88.00.02-2001.

The standards referenced within this document may contain provisions that, through reference in this text, constitute requirements of this document. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this document are encouraged to investigate the possibility of applying the most recent editions of the standards indicated within this document. Members of IEC and ISO maintain registers of currently valid International Standards. ANSI maintain registers of currently valid U.S. National Standards.

This document has been prepared as part of the service of ISA—The Instrumentation, Systems, and Automation Society, toward a goal of uniformity in the field of instrumentation. To be of real value, this document should not be static, but should be subject to periodic review. Toward this end, the Society welcomes all comments and criticisms and asks that they be addressed to the Secretary, Standards and Practices Board; ISA; 67 Alexander Drive; P. O. Box 12277; Research Triangle Park, NC 27709; Telephone (919) 549-8411; Fax (919) 549-8288; E-mail: standards@isa.org.

The ISA Standards and Practices Department is aware of the growing need for attention to the metric system of units in general, and the International System of Units (SI) in particular, in the preparation of instrumentation standards. The Department is further aware of the benefits to USA users of ISA standards of incorporating suitable references to the SI (and the metric system) in their business and professional dealings with other countries. Toward this end, this Department will endeavor to introduce SI-acceptable metric units in all new and revised standards, recommended practices, and technical reports to the greatest extent possible. *Standard for Use of the International System of Units (SI): The Modern Metric System*, published by the American Society for Testing & Materials as IEEE/ASTM SI 10-97, and future revisions, will be the reference guide for definitions, symbols, abbreviations, and conversion factors.

It is the policy of ISA to encourage and welcome the participation of all concerned individuals and interests in the development of ISA standards, recommended practices, and technical reports. Participation in the ISA standards-making process by an individual in no way constitutes endorsement by the employer of that individual, of ISA, or of any of the standards, recommended practices, and technical reports that ISA develops.

CAUTION — ISA ADHERES TO THE POLICY OF THE AMERICAN NATIONAL STANDARDS INSTITUTE WITH REGARD TO PATENTS. IF ISA IS INFORMED OF AN EXISTING PATENT THAT IS REQUIRED FOR USE OF THE STANDARD, IT WILL REQUIRE THE OWNER OF THE PATENT TO EITHER GRANT A ROYALTY-FREE LICENSE FOR USE OF THE PATENT BY USERS COMPLYING WITH THE STANDARD OR A LICENSE ON REASONABLE TERMS AND CONDITIONS THAT ARE FREE FROM UNFAIR DISCRIMINATION.

EVEN IF ISA IS UNAWARE OF ANY PATENT COVERING THIS STANDARD, THE USER IS CAUTIONED THAT IMPLEMENTATION OF THE STANDARD MAY REQUIRE USE OF TECHNIQUES, PROCESSES, OR MATERIALS COVERED BY PATENT RIGHTS. ISA TAKES NO POSITION ON THE EXISTENCE OR VALIDITY OF ANY PATENT RIGHTS THAT MAY BE INVOLVED IN IMPLEMENTING THE STANDARD. ISA IS NOT RESPONSIBLE FOR IDENTIFYING ALL PATENTS THAT MAY REQUIRE A LICENSE BEFORE IMPLEMENTATION OF THE STANDARD OR FOR INVESTIGATING THE VALIDITY OR SCOPE OF ANY PATENTS BROUGHT TO ITS ATTENTION. THE USER SHOULD CAREFULLY INVESTIGATE RELEVANT PATENTS BEFORE USING THE STANDARD FOR THE USER'S INTENDED APPLICATION.

HOWEVER, ISA ASKS THAT ANYONE REVIEWING THIS STANDARD WHO IS AWARE OF ANY PATENTS THAT MAY IMPACT IMPLEMENTATION OF THE STANDARD NOTIFY THE ISA STANDARDS AND PRACTICES DEPARTMENT OF THE PATENT AND ITS OWNER.

ADDITIONALLY, THE USE OF THIS STANDARD MAY INVOLVE HAZARDOUS MATERIALS, OPERATIONS OR EQUIPMENT. THE STANDARD CANNOT ANTICIPATE ALL POSSIBLE APPLICATIONS OR ADDRESS ALL POSSIBLE SAFETY ISSUES ASSOCIATED WITH USE IN HAZARDOUS CONDITIONS. THE USER OF THIS STANDARD MUST EXERCISE SOUND PROFESSIONAL JUDGMENT CONCERNING ITS USE AND APPLICABILITY UNDER THE USER'S PARTICULAR CIRCUMSTANCES. THE USER MUST ALSO CONSIDER THE APPLICABILITY OF ANY GOVERNMENTAL REGULATORY LIMITATIONS AND ESTABLISHED SAFETY AND HEALTH PRACTICES BEFORE IMPLEMENTING THIS STANDARD.

This standard is structured to follow IEC (International Electrotechnical Commission) guidelines. Therefore, the first three clauses discuss the *Scope* of the standard, *Normative References*, and *Definitions*, in that order.

Clause 4 is entitled *Data Model*. The intent of this clause is to describe a data structure for batch control systems using an object model approach.

Clause 5 is entitled *Relational Tables for Information Exchange*. The intent of this clause is to discuss a data format that can be used to share recipes and other batch information across different systems.

Clause 6 is entitled *Procedure Function Charts*. The intent of this clause is to describe a symbolic language for recipe depiction.

This standard is intended for people who are

- a) involved in designing and/or operating batch manufacturing plants;
- b) responsible for specifying controls and the associated application programs for batch manufacturing plants; or
- c) involved in the design and marketing of products in the area of batch control.

The following people served as active members of ISA CommitteeSP88:

NAME	COMPANY
L. Craig, Chairman	Rohm and Haas Co.
T. Fisher, Editor	The Lubrizol Corp.
M. Albano	Honeywell, Inc.
J. Barrault	Siemens AG
D. Brandl	Sequencia Corp.
B. Braunstein	Exxon Chemical Co.
E. Bristol	The Foxboro Co.
L. Charpentier	GSE Systems Inc.
T. Crowl	Siemens Moore Process Automation
C. Eaves	Intellution, Inc.
D. Emerson	Yokogawa Corp. of America
L. Falkenau	E.I. du Pont de Nemours and Co.
M. Fersky	ABB Automation Inc.
D. Fleming	Rockwell Automation
A. Ghosh	ARC

R. Hall	Sequencia Corp.
W. Hawkins	HLQ Ltd.
N. Haxthausen	Novo Nordisk Engineering
B. Jensen	Yokogawa Corp. of America
B. Korkmaz	Automation Vision Inc.
D. Kraska	Geon Co.
T. McFarlane	Neles Automation Inc.
R. Mergen	The Lubrizol Corp.
D. Moffatt	Wonderware Corp.
P. Moylan	Rockwell Automation
T. Müller-Heinzerling	Siemens AG
L. Natiello	Merck
P. Nelson	Dow Corning Corp.
L. Noble	ABB Automation, Inc.
L. Pillai	Pharmacia & Upjohn, Inc.
S. Prichard	GE Fanuc Automation
T. Province	Eastman Chemical Co.
R. Salisbury	ABB Industrial Systems Inc.
M. Saucier	Sequencia Corp.
D. Sweeny	PharmComm Inc.
B. Sykes	Yokogawa Corp. of America
T. Tom	Siemens Moore Process Automation
K. Unger	Real Enterprise Solutions
M. Van Epps	Pharmacia Corp.
J. Vieille	Jean Vieille Conseil
A. Webster	E.I. du Pont de Nemours and Co.
A. Weidenbach	Eastman Chemical Co.
R. Winslow	Sterling Electronics
B. Winters	Honeywell IAC
D. Wolin	Siemens Moore Process Automation

This standard was approved for publication by the ISA Standards and Practices Board on 03 January 2001.

NAME

COMPANY

M. Zielinski	Fisher-Rosemount Systems, Inc.
D. Bishop	Consultant
M. Cohen	Senior Flexonics, Inc.
M. Coppler	Ametek, Inc.
B. Dumortier	Schneider Electric SA
W. Holland	Southern Company
E. Iccayan	Advanced Control & Engineering Solutions
A. Iverson	Ivy Optiks
R. Jones	Dow Chemical Co.
V. Maggioli	Feltronics Corp.
T. McAviney	Bateman Engineering, Inc.
A. McCauley, Jr.	Chagrin Valley Controls, Inc.
G. McFarland	Westinghouse Process Control Inc.
D. Rapley	Rapley Consulting Inc.
R. Reimer	Rockwell Automation
J. Rennie	Factory Mutual Research Corp.
H. Sasajima	Yamatake Corp.
I. Verhappen	Syncrude Canada Ltd.

R. Webb
W. Weidman
J. Weiss
M. Widmeyer
R. Wiegler
C. Williams
G. Wood

Altran Corp.
Parsons Energy & Chemicals Group
EPRI
EG&G Defense Materials
CANUS Corp.
Eastman Kodak Co.
Graeme Wood Consulting

Contents

1	Scope	13
2	Normative references	13
3	Definitions	13
4	Data model	14
4.1	Introduction	15
4.2	Overview model	15
4.3	Recipe model	17
4.4	Equipment model	29
4.5	Production planning and scheduling	31
4.6	Production information management	34
5	Relational tables for information exchange	37
5.1	Introduction	37
5.2	Master recipe information.....	52
5.3	Process cell equipment model exchange.....	64
5.4	Schedule information exchange.....	71
5.5	Production information exchange	75
5.6	Exchange table domains.....	78
6	Procedure function charts.....	80
6.1	Procedure function chart notation	81
6.2	Control recipe depiction	99
6.3	Exception handling.....	99
	Annex A (normative) — Data modeling technique	101
	Annex B (normative) — SQL definition listing	105
	Annex C (informative) — Abbreviations	117
	Annex D (informative) — Language guidelines	119
	Annex E (informative) — Procedure function chart processing examples	121

This page intentionally left blank.

Foreword

- 1) The formal decisions or agreements of the IEC on technical matters, prepared by technical committees on which all the National Committees having a special interest therein are represented, express, as nearly as possible, an international consensus of opinion on the subjects dealt with.
- 2) They have the form of recommendations for international use and they are accepted by the National Committees in that sense.
- 3) In order to promote international unification, the IEC expresses the wish that all National Committees should adopt the text of the IEC recommendation for their national rules insofar as national conditions will permit. Any divergence between the IEC recommendation and the corresponding national rules should, as far as possible, be clearly indicated in the latter.
- 4) The IEC has not laid down any procedure concerning marking as an indication of approval and has no responsibility when an item of equipment is declared to comply with one of its recommendations.

This standard has been prepared by IEC SC65A WG11 and ISA SP88.

It forms Part 2 of a series, the other part being ANSI/ISA-88.01-1995, Batch Control Part 1: Models and Terminology.

Annexes A and B form an integral part of this standard. Refer to annex A for an explanation of the UML notation that is used in this standard. Refer to annex B for a summation of all of the SQL definitions from clause 5. Annexes C and D are for information only.

This page intentionally left blank.

Introduction

ANSI/ISA-88.01-1995, Batch Control Part 1: Models and Terminology (referred to as Part 1 throughout this standard), provides models and terminology applicable to batch control. This standard (referred to as Part 2 throughout the standard) addresses data structures and guidelines for languages. Data structures are addressed by the data model that is defined in clause 4 that more precisely identifies objects and relationships that are addressed by models and concepts of Part 1. Data structures are also addressed by relational tables for information exchange that are defined in clause 5. Languages are addressed by a recipe depiction methodology that is defined in clause 6.

The intended use of the data model is to provide a starting point for developing interface specifications for software components that address any subset of the Part 1 standard. The data model addresses all of the Part 1 standard as an integrated object model, but it does not presume or preclude any specific system architecture or information exchange. The model does not assume any specific division of functionality between systems.

A specific method for the exchange of selected data is defined in clause 5. Relational tables are used as the information exchange method because, within the bounds of the information treated, they

- a) utilize broadly available technologies;
- b) are amenable to translation to other technologies;
- c) are adequate; and
- d) are consistent with other sections of the standard.

Multiple methods of information transfer have not been defined, nor has there been an attempt to identify all information that might be exchanged. In the future, additional methods may be defined to provide alternate ways to exchange data.

Clause 6 defines the symbols and rules for a graphical language that can be used to depict recipes. Recipes are the central feature of batch control, and they can address a wide range of complexity, but there is no one depiction that is ideal for all circumstances. A simple table, for example, might be the most appropriate recipe form for simple cases. This standard specifies a method for depiction of master and control recipe procedures that can be applied over a broader range of complexity.

Although this standard is intended primarily for batch processes, there may be considerable value for other types of processes.

This page intentionally left blank.

1 Scope

This Part 2 standard on Batch Control defines data models that describe batch control as applied in the process industries, data structures for facilitating communications within and between batch control implementations, and language guidelines for representing recipes.

2 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions this standard. At the time of publication, the editions indicated were valid. All normative documents are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. Members of IEC and ISO maintain registers of currently valid normative documents.

IEC 60848:1988, *Preparation of function charts for control systems*

IEC 60902:1987, *Industrial-process measurement and control: Terms and definitions*

NOTE — IEC 60902:1987 has been replaced by IEC 60050-351:1998, International electrotechnical vocabulary-Part 351: Automatic control

IEC 61131-3:1993, *Programmable controllers – Part 3: Programming languages*

IEC 61512-1:1997, *Batch control – Part 1: Models and terminology*

ANSI/ISA-88.01-1995, *Batch Control – Part 1: Models and Terminology*

ISO/IEC 9075:1992, *Information processing systems – Database language SQL with integrity enhancement*

3 Definitions

ISA hereby grants a non-exclusive, royalty-free, limited license under ISA's copyright in the standard, to copy, display and distribute this section of this ISA standard (including software included in or defined by such section), as follows:

1. Producers of products or services intended to comply with the standard may incorporate this designated section, but only to the extent reasonably necessary to make, use, and distribute any product or service (including product documentation) that is compliant with this standard.
2. End users of a product or service made by a producer acting under the preceding license may reproduce and use the designated section, but only to the extent reasonably necessary to enjoy the intended functions of the product or service and to maintain, configure, or reconfigure systems to be compliant.
3. Persons providing education on or promotion of the standard may copy, display and distribute the designated section, but only to the extent reasonably necessary to provide information related to the standard.

Except as expressly permitted, all other reproduction and distribution without permission of ISA is prohibited. All copies of this section of the standard made or distributed under this license must cite the standard and include the following notice of copyright:

Copyright © 2001 by ISA—The Instrumentation, Systems, and Automation Society. All rights reserved. Used with permission of ISA.

For the purposes of this standard, the following definitions apply. Definitions and concepts expressed in ANSI/ISA-88.01-1995 apply, except where differences are explicitly stated in this Part 2 standard. Definitions in IEC 60902:1987 were also used as a basis.

3.1 allocation symbol:

a graphical symbol that is used to represent the encapsulation of the resource allocation and de-allocation rules for a recipe procedural element.

3.2 building block:

a recipe entity that exists in a library.

3.3 enumeration set:

a list of predefined strings and their associated numerical values.

3.4 exchange table:

a database table that is used to exchange batch-related information between systems.

3.5 link:

an object that specifies the connection between two other objects (e.g., the connection between recipe entities or between recipe entities and transitions).

3.6 procedure function chart:

a graphical representation of a recipe procedure that specifies the processing order for recipe procedural elements.

3.7 recipe element:

a structural entity that is used to represent recipe entities and symbols, except transitions and directed links, that are used in procedure function charts.

3.8 recipe entity:

the combination of a procedural element with associated recipe information (e.g., header, formula, equipment requirements, other information). General, site, master, and control recipes are also recipe entities.

4 Data model

ISA hereby grants a non-exclusive, royalty-free, limited license under ISA's copyright in the standard, to copy, display and distribute this section of this ISA standard (including software included in or defined by such section), as follows:

- 1. Producers of products or services intended to comply with the standard may incorporate this designated section, but only to the extent reasonably necessary to make, use, and distribute any product or service (including product documentation) that is compliant with this standard.**
- 2. End users of a product or service made by a producer acting under the preceding license may reproduce and use the designated section, but only to the extent reasonably necessary to enjoy the intended functions of the product or service and to maintain, configure, or reconfigure systems to be compliant.**

- 3. Persons providing education on or promotion of the standard may copy, display and distribute the designated section, but only to the extent reasonably necessary to provide information related to the standard.**

Except as expressly permitted, all other reproduction and distribution without permission of ISA is prohibited. All copies of this section of the standard made or distributed under this license must cite the standard and include the following notice of copyright:

Copyright © 2001 by ISA–The Instrumentation, Systems, and Automation Society. All rights reserved. Used with permission of ISA.

4.1 Introduction

This clause defines data models that specify a set of objects, attributes, and their basic relationships that cover the concepts of Part 1 at a high level of abstraction. The models apply to interfaces to batch control systems in a technology-independent manner. The models are not intended to address the internal system architecture of batch control systems.

The intended use of these models is to provide a starting point for developing interface specifications for software components that address any subset of Part 1.

The models address all of Part 1 as an integrated object model, but they do not presume or preclude any specific system architecture or information exchange. The models do not assume any specific division of functionality between systems.

In the cases where the objects and relationships defined in this clause are presented through an interface, then that interface shall use the object names, the attribute names, and the relationships of this clause commensurate with the interface technology chosen and the capabilities offered. An example of such an interface is the SQL relational table interface that is defined in clause 5.

An exchange format or interface specification may implement only some objects or parts of objects (e.g., not all properties are defined). An exchange format or interface specification may also provide additional objects or properties (e.g., phase duration information), including the expansion of any data model attribute into multiple attributes. Any such implementation shall be consistent with the data model that is presented here and the concepts of Part 1.

4.1.1 Modeling techniques

The models that are described in this clause are based on the Unified Modeling Language (UML) (see clause A.1).

The tables describe only the class attributes of the objects. The relationships between objects are described in the figures.

4.2 Overview model

This model (see figure 1) gives a high level overview of the main classes that are defined and the relations between these classes for the batch domain that is described by the Control Activity Model of Part 1. The individual object classes are more thoroughly described in the subsequent models in this subclass.

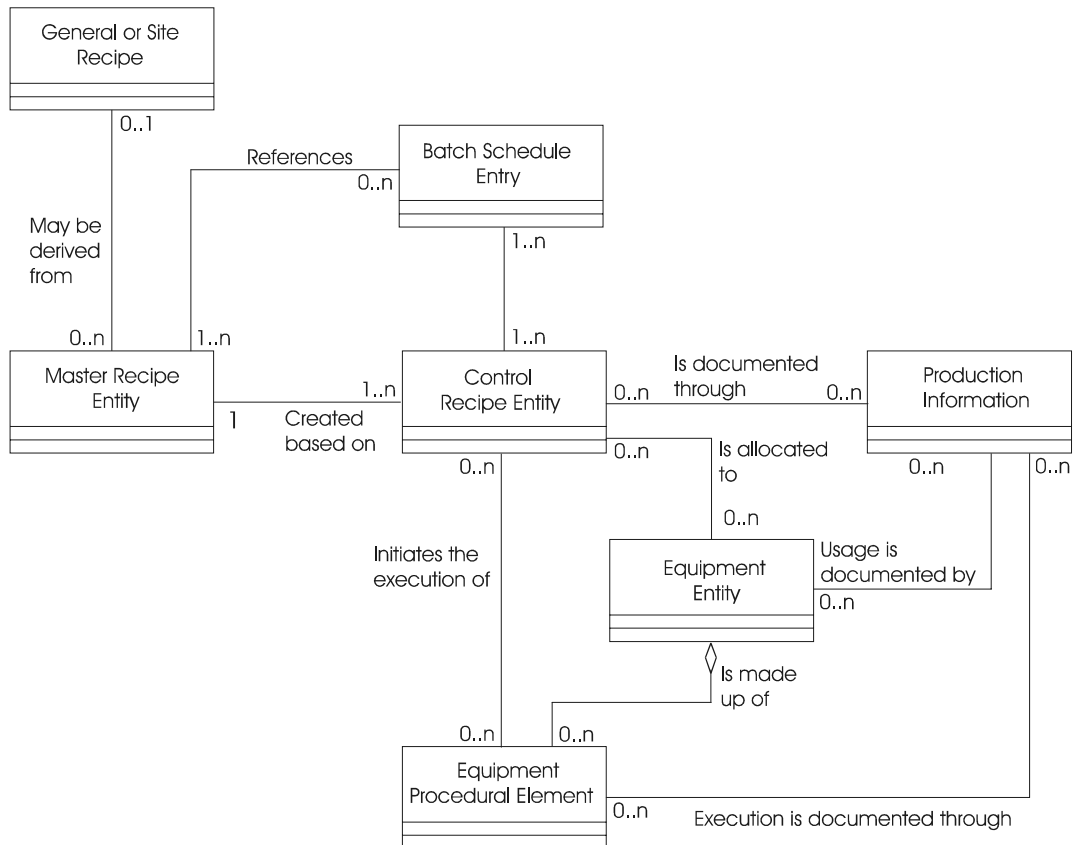


Figure 1 — Overview model

A general or site recipe is made up of a hierarchy of *general recipe entities* that correspond to the procedural entities (process stages, process operations, and process actions).

A master recipe may be derived from a general or site recipe. The master recipe itself may be seen as a top-level master recipe entity. A master recipe is made up of a hierarchy of *master recipe entities* that correspond to the procedural entities (i.e., procedures, unit procedures, operations, phases).

A *batch schedule entry* specifies production of a specific batch through the execution of a recipe. A batch schedule is, in principle, a list that specifies the production of batches, and it includes information about timing. The product-specific information necessary for this purpose is derived from a related *master recipe entity*.

Based on a *Batch Schedule Entry*, a control recipe starts as a copy of a specific version of a master recipe, and it is then modified to create the recipe that will produce the batch. A control recipe includes the information necessary for equipment control.

Control recipe entities are created based on *master recipe entities*. A control recipe may be enhanced with additional information (e.g., scaling, equipment assignment), and it may be modified (including creating or removing control recipe entities).

Equipment entities are selected and allocated to the *control recipe entities*.

A *control recipe entity* may be linked to an *equipment procedural entity* within the *equipment entity* (normally the unit). The *equipment procedural entity* may be initiated, and its parameters may be assigned recipe values.

Production information is generated during the production of the batch. This information may be related to recipe entities, equipment entities, and/or equipment procedural elements.

4.3 Recipe model

4.3.1 Recipe entity

Recipes are organized hierarchically, with various categories of information at each level. The recipe entity is the construct that is used to represent the tight coupling of the data at a particular level.

The recipe entity is the fundamental structure in all kinds of recipes (see figure 2). The recipe entity structurally takes the place of the recipe procedural element as defined in Part 1, but it may include any or all of the recipe components: procedural definitions, parameters with values, equipment requirements and other information.

The class specifications are shown in table 1.

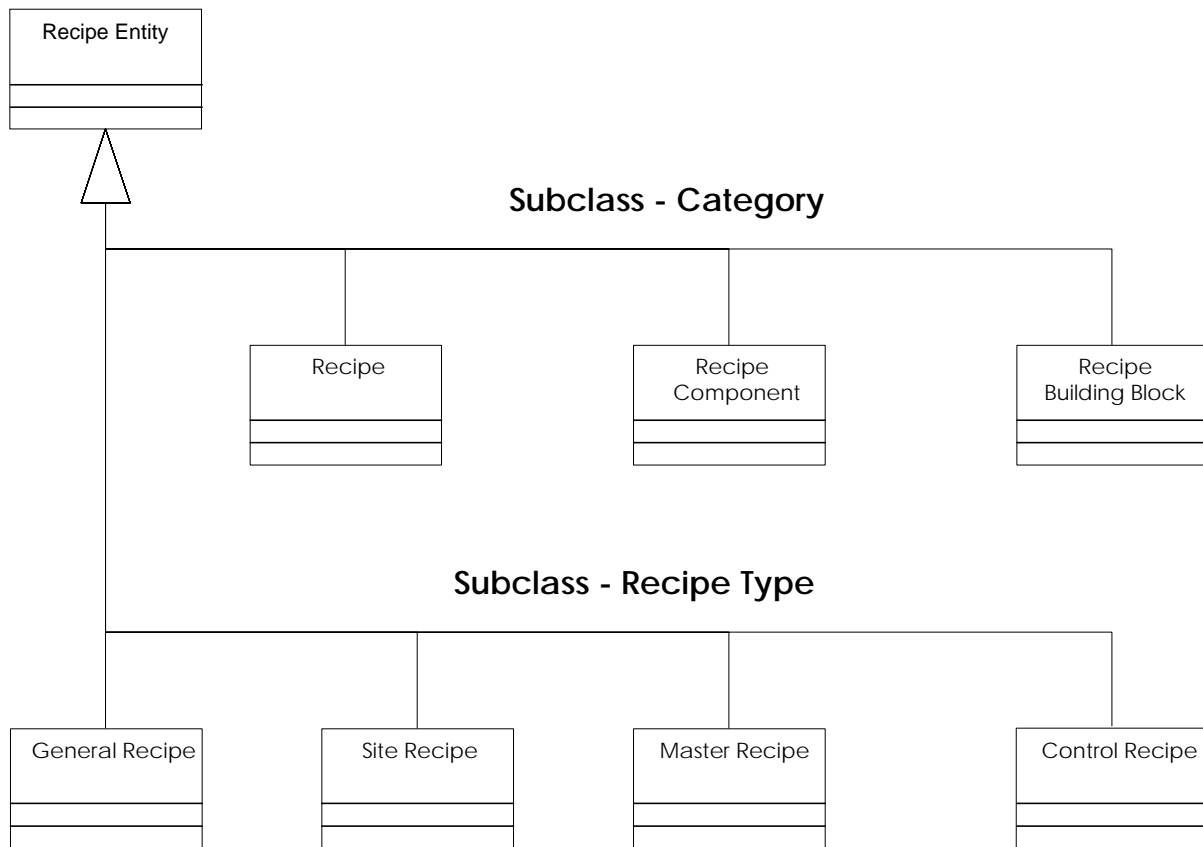


Figure 2 — Recipe entities

Table 1 — Recipe entity

NAME	RECIPE ENTITY
FunctionalDescription	A recipe entity is the combination of a procedural element with associated recipe information (e.g., header, formula, equipment requirements and other information). General, site, master, and control recipes are recipe entities. One example is a unit recipe (ANSI/ISA-88.01-1995, 5.3.2). "unit recipe: the part of a control recipe that uniquely defines the contiguous production requirements for a unit." (ANSI/ISA-88.01-1995, 3.62).
ATTRIBUTES	
RecipeEntityID	Provides unique Identification.

A recipe itself is considered to be a recipe entity (category: recipe). The recipe is built up of lower-level recipe entities (e.g., unit recipes) (category: component). When constructing a specific recipe, the components may be created from library elements (category: building block).

The recipe entity concept is applied to all recipe types: general, site, master and control. When a recipe is executed, the representation in the batch history of the executed recipe entities will have much the same structure, and it is therefore shown as a subclass. An overview of the subclasses is shown in table 2. Subclasses by category are shown in tables 3 to 5. Subclasses by type are shown in tables 6 to 9. General and site recipes are not discussed further in this subclass.

Table 2 — Subclasses – overview

	GENERAL RECIPE ENTITY	SITE RECIPE ENTITY	MASTER RECIPE ENTITY	CONTROL RECIPE ENTITY
RECIPE	A complete and self-contained general recipe.	A complete and self-contained site recipe.	A complete and self-contained master recipe.	A complete and self-contained control recipe.
RECIPE BUILDING BLOCK	A generic general recipe entity type that can be instantiated in a specific recipe or in another building block.	<i>Building blocks for site recipes may not exist. Site recipes are normally modified using general recipe building blocks.</i>	A generic master recipe entity type that can be instantiated in a specific recipe or in another building block.	<i>Building blocks for control recipes do not exist. Control recipes are modified using master recipe building blocks.</i>
RECIPE COMPONENT	A component of a general recipe or library element - may be an instantiation of a building block.	A component of a site recipe or library element - may be an instantiation of a general recipe building block.	A component of a master recipe or library element - may be an instantiation of a building block.	A component of a control recipe - may be an instantiation of a master recipe building block.

Table 3 — Recipe

NAME	RECIPE
FunctionalDescription	The top level recipe entity.
ATTRIBUTES	
RecipeID	Identifies the recipe. When combined with the "RecipeVersion," defines a unique instance of a recipe.
RecipeVersion	Identifies the version of a recipe. When combined with a "RecipeID," defines a unique instance of a recipe (e.g., Red Oak - A10.3).
VersionDate*	Identifies the date and time that this version of the recipe was created or modified.
ApprovalDate*	Identifies the date and time that this version of the recipe was approved.
EffectiveDate*	Identifies the earliest date and time that this version of the recipe may be used.
ExpirationDate*	Identifies the date and time that this version of the recipe expires.
ProductID*	Identifies the product or product family that would be created by execution of this version of the recipe (e.g., Premium Beer).
Author*	Identifies the person or system that authored this version of the recipe. (e.g., J. Smith).
ApprovedBy*	Identifies the person or system that approved this version of the recipe.
Description	Describes this version of the recipe and/or product (e.g., North Carolina's Finest Premium Beer).
Status*	Defines the Status of the information (e.g., "Approved for Production", "Approved for Test", "Not Approved", "Inactive", "Obsolete").
*Not required for control recipe (available by reference to master recipe).	

Table 4 — Recipe component

NAME	RECIPE COMPONENT
FunctionalDescription	A recipe entity that is part of a recipe or building block (i.e., an instance of a building block in a particular recipe or containing building block recipe entity).
ATTRIBUTES	
Level	Indicates the procedural hierarchy level (i.e., process stage, process operation, or process action for general and site recipes and unit procedure, operation, or phase for master and control recipes).
RE_Use	Defines if the recipe component is a copy of a building block or a reference to it.

Table 5 — Recipe building block

NAME	RECIPE BUILDING BLOCK
FunctionalDescription	A recipe entity that exists in a library. A building block can be parameterized and used when building recipes.
ATTRIBUTES	
RecipeVersion	Identifies the version of the recipe entity.
VersionDate	Identifies the date and time that this version of the recipe was created or modified.
ApprovalDate	Identifies the date and time that this version of the recipe was approved.
Author	Identifies the person or system that authored this version of the recipe (e.g., J. Smith).
ApprovedBy	Identifies the person or system that approved this version of the recipe.
Description	Describes the function that is achieved through execution of this version of the recipe entity.
Level	Indicates the level of the recipe entity.
UsageConstraint	Defines other rules that determine the usage (e.g., "always succeeded by..." or "never runs in parallel with...").
Status	Defines the Status of the recipe entity (e.g., "Approved for Production", "Approved for Test", "Not Approved", "Inactive", "Obsolete").
Function	Defines how the recipe entity is executed (e.g., by referencing an equipment procedural element, through execution of contained logic).

Table 6 — General recipe entity

NAME	GENERAL RECIPE ENTITY
FunctionalDescription	All of a general or site recipe, a component of a general or site recipe, or a building block for creation of a general or site recipe.
ATTRIBUTES	
ScaleReference	Defines a reference scale for the parameter values.

Table 7 — Site recipe entity

NAME	SITE RECIPE ENTITY
FunctionalDescription	All of a site recipe, a component of a site recipe, or a building block for creation of a site recipe.
ATTRIBUTES	
ScaleReference	Defines a reference scale for the parameter values.

Table 8 — Master recipe entity

NAME	MASTER RECIPE ENTITY
FunctionalDescription	All of a master recipe, a component of a master recipe, or a building block for creation of a master recipe.
ATTRIBUTES	
ScaleReference	Defines a reference scale for the parameter values.
ProcessCellID	Identifies the equipment category for which the recipe entity was defined (e.g., process cell or process cells for which this master recipe was defined).

Table 9 — Control recipe entity

NAME	CONTROL RECIPE ENTITY
FunctionalDescription	A recipe entity that is all of or a part of a control recipe.
ATTRIBUTES	
BatchID	Specifies the actual Batch ID.
BatchSize	Defines the requested size or scale factor for the batch, based on the scale factor for the batch as defined in the master recipe.
Status	Describes execution status (e.g., not yet activated, active, or completed).

4.3.2 Parts of recipe entities

The model (see figure 3 and tables 10 through 13) shows the categories of information of a recipe as specified in Part 1. The model indicates that these components may exist at any level of the decomposition of the recipe (e.g., a unit recipe may have its own equipment requirements).

- a) The Header category of information is contained in attributes to the recipe entity itself, instead of being a distinct object class in this model.
- b) The Formula category of Part 1 is modeled as a set of parameter objects. All levels of the recipe decomposition may have parameters, including the recipe itself. See 4.3.6 on parameters.
- c) The modeling of Equipment Requirements is discussed in 4.3.5.
- d) The Other Information category, as defined in Part 1, is represented as a single object class, even though other information may have multiple elements and different structures.
- e) The Procedure category of Part 1 is modeled as a set of procedural structural elements.

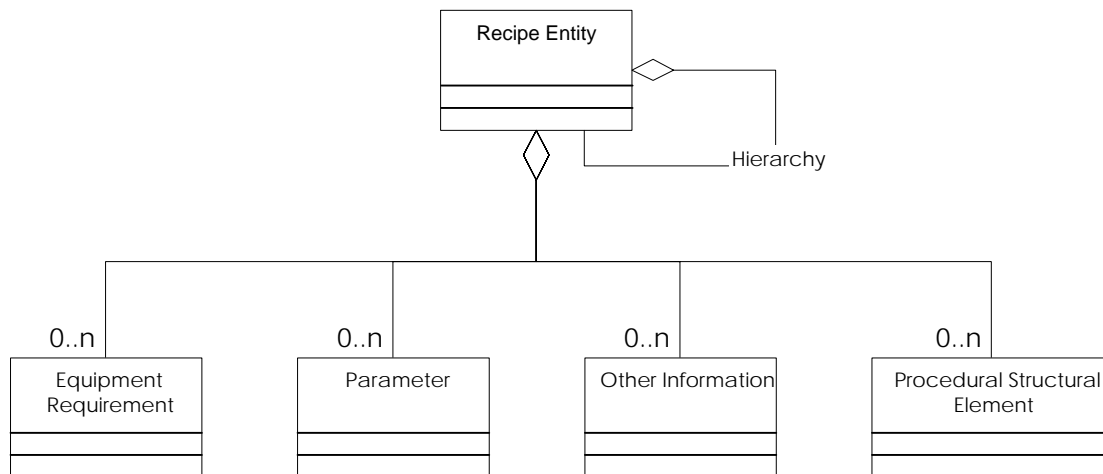


Figure 3 — Parts of recipe entities

Table 10 — Parameter

NAME	PARAMETERS
FunctionalDescription	Formula values or placeholders for values that are to be communicated to and from recipe entities during execution.
ATTRIBUTES	
ParameterID	Provides unique identification.
ParameterType	Specifies how the parameter value is interpreted (e.g., constant, reference equation).
Description	Describes the parameter or use of the parameter.
EngineeringUnits	Identifies the engineering units of measure for the Value (e.g., kg, pounds).
Value	Contains the parameter value. If Value is a relation, it contains the equation, form, deferral rule or whatever ties the related parameters together. If Value is a building block parameter, this attribute will hold the default value.
Scaled	Specifies the scaling rule. Simplest case: scaled or not scaled with batch reference size.
Usage	Specifies the parameter as a process input, process output, or process parameter.

Table 11 — Equipment requirement

NAME	EQUIPMENT REQUIREMENT
FunctionalDescription	Represents an equipment requirement as specified in the recipe entity.
ATTRIBUTES	

Table 12 — Other information

NAME	OTHER INFORMATION
FunctionalDescription	A category of recipe information that may contain batch processing support information that is not contained in other parts of the recipe (e.g., regulatory compliance information, materials and process safety information, process flow diagrams, packaging/labeling information).
ATTRIBUTES	

Table 13 — Procedural structural element

NAME	PROCEDURAL STRUCTURAL ELEMENT
FunctionalDescription	The recipe procedural elements and the ordering information for their execution.
ATTRIBUTES	

4.3.3 Recipe entity relation (procedural structure)

Recipe entities are decomposed hierarchically along the structures for procedural entities as defined in Part 1 (i.e., a recipe procedure contains unit procedures that contain operations that contain phases). This hierarchy is modeled using recursive containment. Higher-level objects may contain lower-level objects.

Procedural structural elements include the recipe procedural elements and the connections (e.g., link, transition) that are used to order them (e.g., a unit recipe’s procedural structural elements are the operations and the ordering of those operations contained within it). The procedural structural elements may be related to other procedural structural elements.

4.3.4 Recipe building blocks

Recipe building blocks are an important concept in the data model (see figure 4). This figure represents the relations at a single level of the procedure hierarchy.

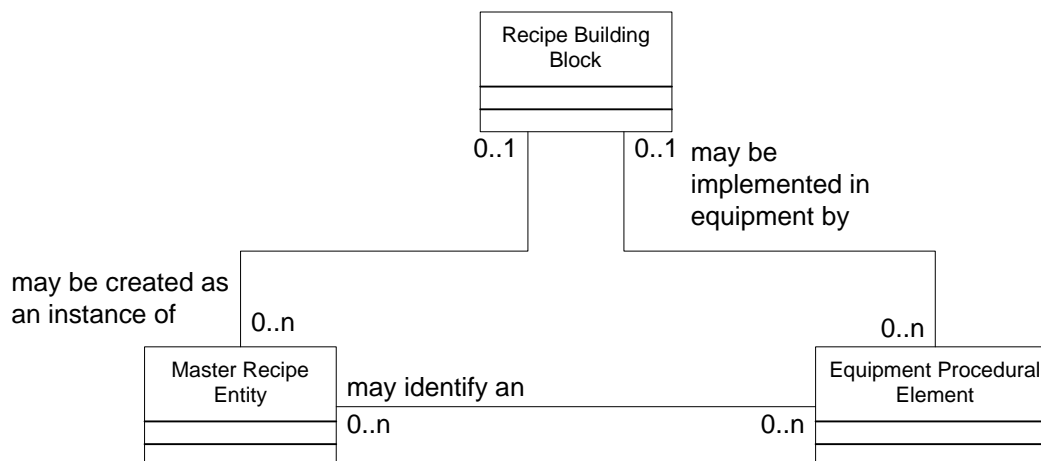


Figure 4 — Recipe building block

Recipe building blocks are the building blocks from which master recipes are created. When a *recipe building block* is instantiated in a master recipe as a *master recipe entity*, it may carry parameters, equipment requirements, and other information that may be assigned master recipe specific values. Lower-level contents of the *recipe building block* (e.g., subordinate recipe entities) may be copied into *master recipe entities*. These same lower-level contents may also be accessible by reference to the *recipe building block*.

A *recipe building block's* functionality may be implemented in equipment through *equipment procedural elements* (see table 14), which is necessary for the execution of the lowest-level recipe entities (i.e., the recipe entities that are intended to be linked to equipment procedural elements).

Table 14 — Equipment procedural element

NAME	EQUIPMENT PROCEDURAL ELEMENT
FunctionalDescription	A procedural element that is associated with a piece of equipment (e.g., an equipment phase or equipment operation).
ATTRIBUTES	
EquipmentProceduralElementID	Provides unique identification.
Version	Identifies the version of the procedural element.
VersionDate	Identifies the date and time that this version was created or modified.
ApprovalDate	Identifies the date and time that this version was approved.
Author	Identifies the person or system that authored this version (e.g., J. Smith).
ApprovedBy	Identifies the person or system that approved this version.
Description	Describes the function that is achieved through execution of the recipe entity.
Level	Indicates the level of the equipment entity. The equipment entity may only be used at this level.
Mode	Indicates the current mode of the procedural element.
State	Indicates the current state of the procedural element.

The mechanisms may be illustrated by the following example (see figure 5) that could represent an object model of a part of an actual application. The generic concept of a building block is instantiated as a specific building block "heat." The "is based on" relation between building blocks and components is replaced by a subclass relation (this is one possible implementation, and it indicates that if the "heat" building block is changed, then the change will propagate to all recipes that use "heat"). Another implementation could be that "heat" is just copied when instantiated.

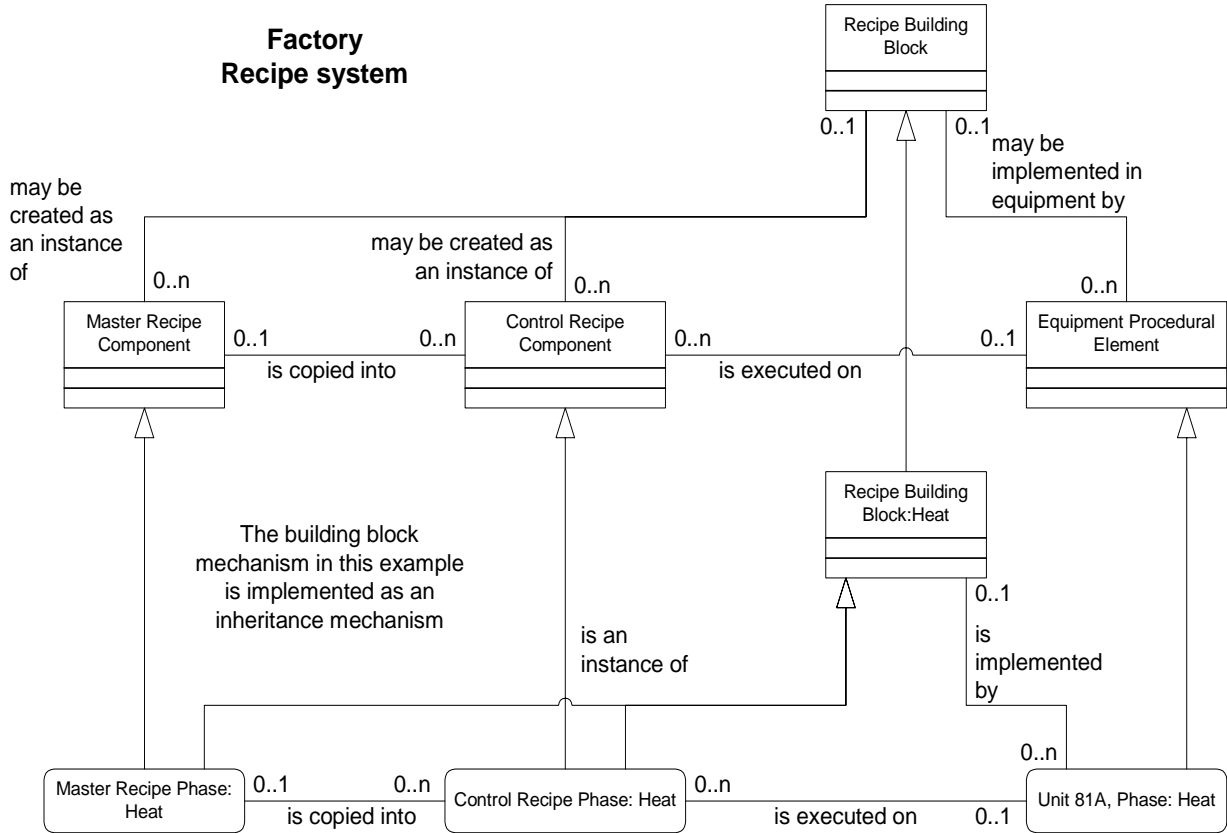


Figure 5 — Building block concept

4.3.5 Equipment requirements

Recipe entities may contain *equipment requirements* (see figure 6 and tables 15, 16, and 17). Equipment requirements reference a specific *equipment property type* (e.g., an *equipment property type* may be the 'size of vessel' or the 'lining of vessel'). A specific equipment requirement could then specify a minimum value for the size of a vessel.

This requirement may then be met by a piece of equipment with a certain equipment property that references the same equipment property type. For example, a specific unit, UNIT12, would have a value for the property type 'size of vessel'.

An equipment entity is a specific piece of equipment, and it may be replaced by a class of equipment (equipment class). See 4.4.1.

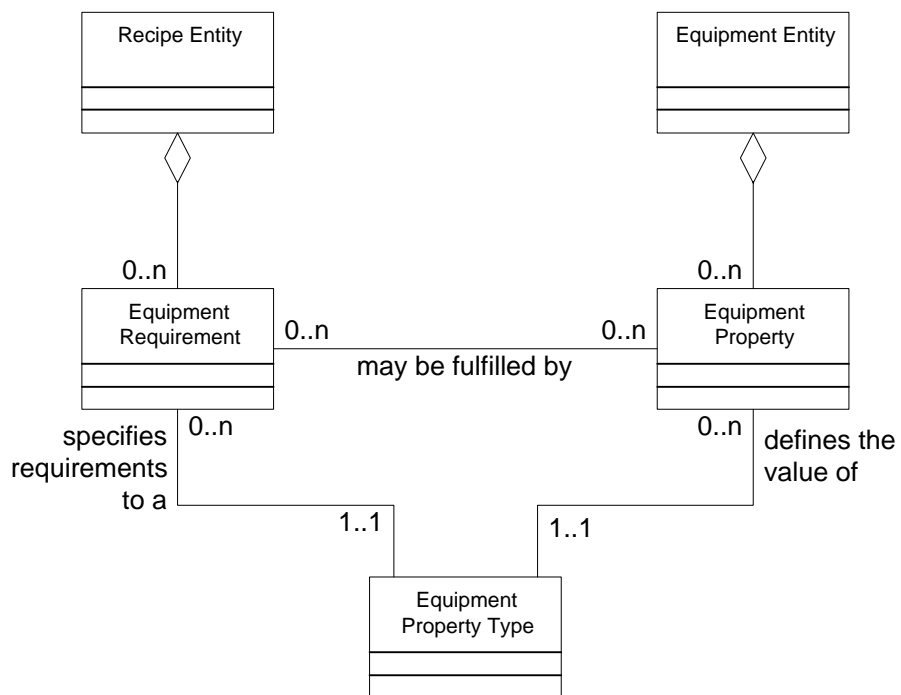


Figure 6 — Recipe entity equipment requirements

Table 15 — Equipment entity

NAME	EQUIPMENT ENTITY
FunctionalDescription	A collection of physical processing and control equipment and equipment control that is grouped together to perform a certain control function or set of control functions.
ATTRIBUTES	
EquipmentEntityID	Provides unique identification.
EquipmentLevel	Specifies the physical hierarchy level (e.g., process cell, unit, equipment module, control module).
Mode	Indicates the current mode of the equipment entity.
State	Indicates the current state of the equipment entity.

Table 16 — Equipment property

NAME	EQUIPMENT PROPERTY
FunctionalDescription	Identifies a property that the equipment entity or class supplies. These properties are application specific (e.g., lining type, size, heat capability, steam temperature).
ATTRIBUTES	
PropertyID	Provides unique identification.
Value	Identifies the value of the property (e.g., glass, 50000, 650).
ValueRange	Defines limits or constraints that are related to Value.
EngineeringUnits	Defines the engineering units of the property.
Description	Describes the type of the equipment property.

Table 17 — Equipment property type

NAME	EQUIPMENT PROPERTY TYPE
FunctionalDescription	The general class of equipment attributes (e.g., lining type, size, heat capability, steam temperature).
ATTRIBUTES	

Control recipe entities will initially contain the equipment requirements that are copied from the *master recipe entity*, and these need to be fulfilled by the corresponding property of one or more equipment entities in order for specific equipment to be allocated. The initial *equipment requirement* may be replaced by specific equipment allocations. These allocations are also modeled as equipment requirements.

4.3.6 Recipe parameters

Parameters are variables associated with recipe entities. These variables may be used by equipment procedural elements, they may be used by other activities (e.g., scheduling), or they may be referenced by other parts of the recipe (e.g., transition criteria) (see figure 7).

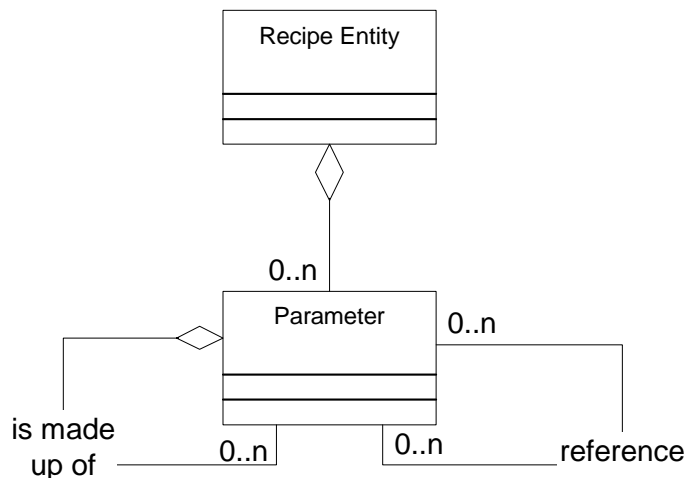


Figure 7 — Parameter model

Parameters may be categorized as process inputs, process outputs, or process parameters.

Parameters may be made up of a collection of other parameters. The model supports the concept of structured parameters. Therefore, the model allows the possibility of including parameters of different types (process parameters, process inputs, process outputs) in the same structure as well as defining single type data structures.

Parameter value attributes may be organized by defining parameter value types. Parameter value types could include

- a) IEC 61131-3 basic data types;
- b) compatibility matrix information that is used for determining clean-in-place (CIP) or sterilize-in-place (SIP) requirements;
- c) data sets that define material transactions (transfer, consumption, generation of material); or
- d) data series (e.g., a temperature profile that will be tracked).

Parameter values may be simple values, expressions, or references to parameters that are defined at the same level or higher levels in the procedural hierarchy. Values that are expressions may include references to other parameters.

Parameters may be related in a number of ways, including the following:

- a) Algebraic or Boolean equations
- b) Product specific entry forms that work on one or more parameters
- c) Standard operating procedures (SOPs) that display or otherwise utilize parameters (e.g., dynamic values, recipe values)
- d) Deferral of parameters to different recipe entities (at the same or another level)

e) External applications that use parameters

The formula is represented in the data model as recipe parameters (see table 10). A recipe’s formula is a collection of selected parameters to the recipe procedure, and it may also include parameters that are defined at lower levels of the procedural hierarchy.

Parameters are often scaled, based on batch size or other key attributes. Scaling may be more complex than a simple linear relationship. More complex scaling methods can be accommodated with user-defined algorithms and relationships.

4.4 Equipment model

The physical structure of the plant needs to be taken into account in the evaluation of equipment selection during recipe execution (see figure 8 and table 18). In particular, the transfer capabilities between equipment or the ability to allocate shared equipment are important to routing a batch.

Equipment entities are defined as in the hierarchy that is specified in Part 1. This hierarchy is modeled through the recursive nature of the objects. This construct allows for the configuration of expandability and collapsibility.

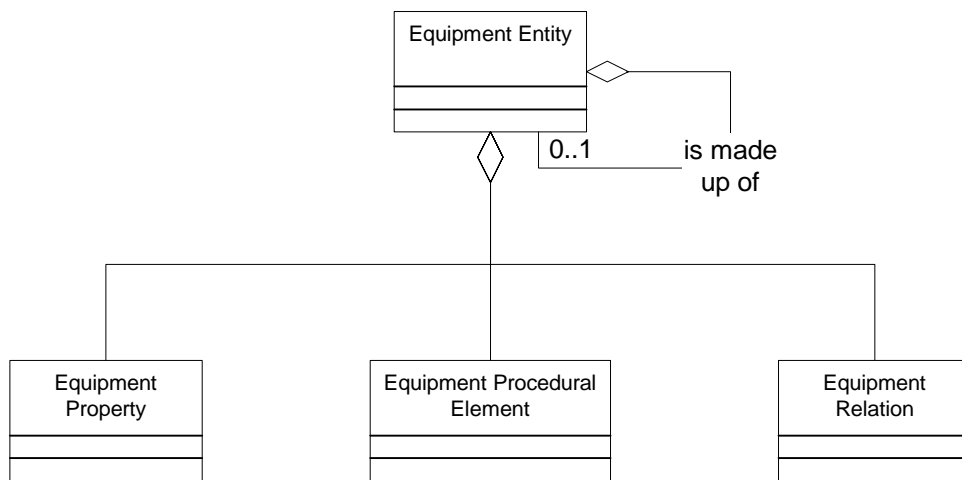


Figure 8 — Equipment structure

Table 18 — Equipment relation

NAME	EQUIPMENT RELATION
FunctionalDescription	A representation of the connections between equipment (e.g., pipes, conveyors, flexible connections), but may also be used to represent other kinds of relations between equipment (e.g., relation to shared-use equipment).
ATTRIBUTES	
RelationID	Provides unique identification.

The equipment in, for example, a process cell (i.e., units, equipment modules, control modules) is normally related to each other by pipes or other connections. The connections may be modeled as equipment relations (see figure 9), optionally with a direction (e.g., a flow direction). The relations (e.g., pipes) are part

of the higher-level equipment entity. The connections may conveniently be categorized in relation classes and this ensures a consistent evaluation. Equipment relations include

- a) permanent connection;
- b) temporary connection;
- c) may be used as resource for; and
- d) always runs the same product as.

Note that relations other than connections may be possible.

Equipment may have properties. The properties are specific to each user implementation. Equipment properties may be used for examining equipment characteristics and matching recipe equipment requirements. See 4.3.5.

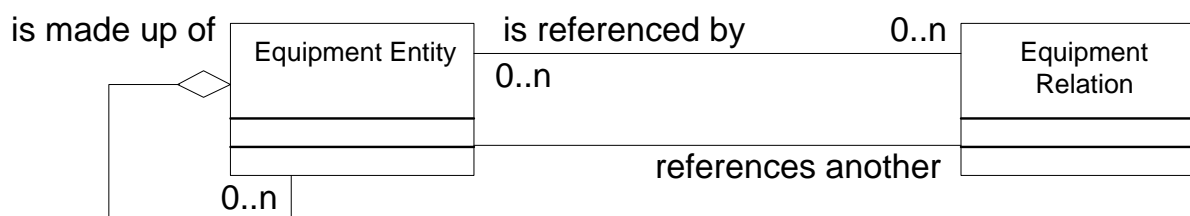


Figure 9 — Equipment entity relations

4.4.1 Equipment classes

Equipment classes (see figure 10 and table 19) provide a means to group equipment entities by common characteristics. Equipment entities may be a member of one or more equipment classes, or they may not belong to a class at all. Equipment classes may be used to specify groups of units and they may be used as alternatives during equipment selection. For example, a recipe may require a reactor for a unit procedure, so its equipment requirements may specify a specific reactor (e.g., R-101), a set of reactor units (e.g., R-101, R-103) or the reactor class (e.g., the class “Reactor” that contains reactors R-101, R-102 and R-103).

Equipment entities may be members of an *equipment class*, and the class determines some of the properties of the class members. As an example, certain equipment properties (e.g., glass lining) are shared with the class.

Equipment entities may belong to zero or more *equipment classes* (e.g., vessel BV1 may be both a reactor and a holding tank).

Equipment classes may determine some or all *equipment properties*, *equipment procedural elements* and *equipment relations* of the referencing equipment entities.

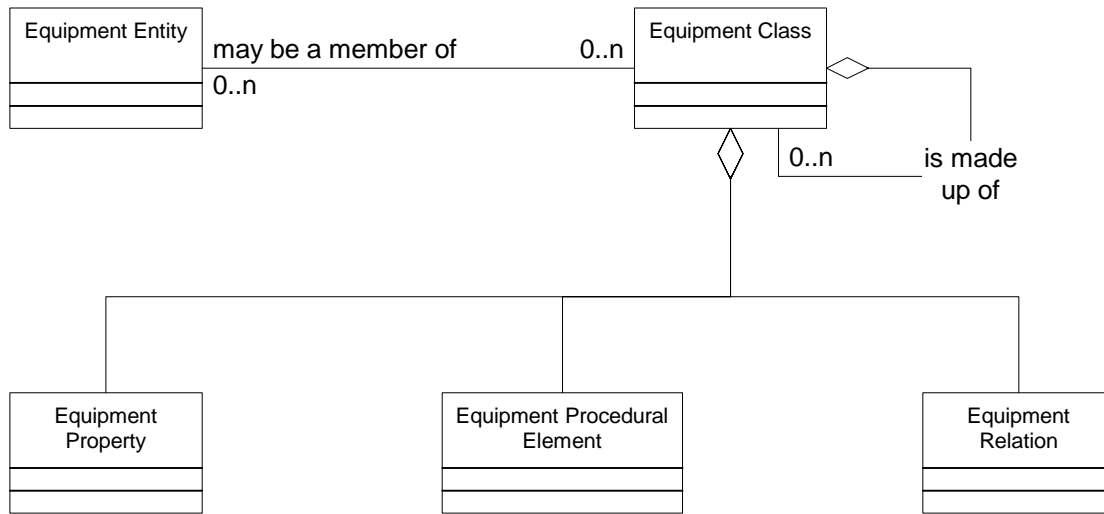


Figure 10 — Equipment classes

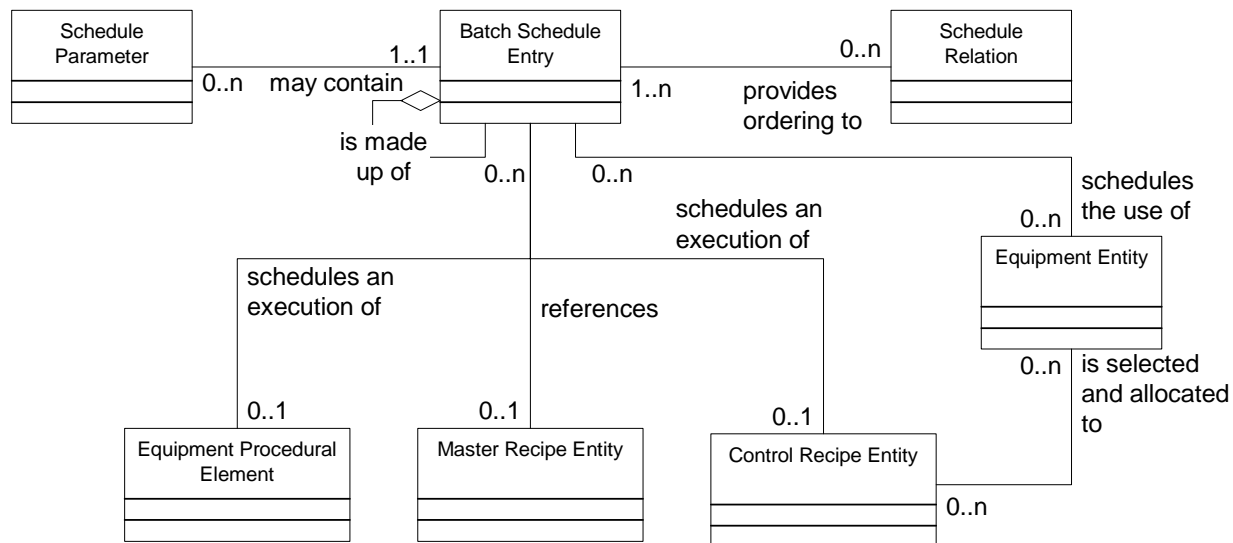
Table 19 — Equipment class

NAME	EQUIPMENT CLASS
FunctionalDescription	An equipment entity class.
ATTRIBUTES	
EquipmentClass	Provides unique identification.
EquipmentLevel	Specifies the physical hierarchy level (e.g., process cell, unit, equipment module, control module).

4.5 Production planning and scheduling

The central entity in a schedule (see figure 11) is the batch schedule entry. This object defines an intended execution of one or more batches/control recipes, or other control recipe entities (typically unit procedures)(see table 20). The batch schedule entry may also be used to schedule other activities (e.g., equipment downtime). A batch schedule entry may include formula/parameter values that are to be used in a control recipe (see table 21).

The batch schedule entry can be used to represent higher-level schedule entities (e.g., a production campaign or a production order).

**Figure 11 — Batch schedule****Table 20 — Batch schedule entry**

NAME	BATCH SCHEDULE ENTRY
FunctionalDescription	A scheduled item that represents a unit procedure in a batch, a complete batch, or a set of batches (e.g., a campaign).
ATTRIBUTES	
ID	Provides unique identification (e.g., actual campaign, lot, batch ID, procedural entity ID).
Level	Specifies the hierarchy level (e.g., campaign, batch, unit procedure).
BatchSize	Defines the requested size or scale factor for the batch, based on the scale factor for the batch as defined in the master recipe.
Schedule	Defines scheduled execution times (start/stop).
ResourceUsage	Specifies resource usage for this schedule entry.
Status	Specifies schedule status (e.g., proposed for evaluation (such as what-if analysis), planned, committed, started, completed).

Table 21 — Schedule parameter

NAME	SCHEDULE PARAMETER
FunctionalDescription	Formula values that are to be communicated to and from batch schedule entries.
ATTRIBUTES	
ParameterID	Provides unique identification.
ParameterType	Includes how the Value is interpreted (e.g., constant, reference, or equation).
Description	Describes the parameter or use of the parameter.
EngineeringUnits	Identifies the engineering units of measure for the Value (e.g., kg, pounds).
Value	Contains the parameter value. If it is a relation, Value contains the equation, form, deferral rule or whatever ties the related parameters together. If Value is a building block parameter, this attribute will hold the default value.
Scaled	Defines the scaling rule. Simplest case: scaled or not scaled with batch reference size.
Usage	Identifies the parameter as a process input, process output, or process parameter.

The schedule relations can be used to represent the scheduling-relevant subset of recipe relations (e.g., the relations related to batch transfers)(see table 22). At higher levels, they can be used to represent required or desirable relations between schedule entries (e.g., batch xx should follow batch yy or a specific cleaning should take place between the two batches). The specific subclasses and characteristics of schedule entry relations are not modeled here.

A higher-level batch schedule entry will include the lower-level schedule entries and relations (e.g., a scheduled campaign may include scheduled batches and relations between these batches).

In the simplest case, the batch schedule entry represents a batch in a batch execution list or queue that is expected to be initiated in sequence. Intended starting times and projected duration/ending times may be added. Further, equipment assignments and the use of other resources may be specified by the batch schedule entry. The scheduling may happen at more detailed levels (e.g., scheduling of individual unit recipes and their assignment to equipment, and potentially scheduling individual operations or phases, their projected duration, and their consumption of resources, including shared or exclusive-use resources that constrain the schedule). Units and equipment modules are allocated or de-allocated as required by the specific control recipe.

The collection of schedule entries may be viewed in the following different ways:

- a) As a list or chart of batches in a process cell or part of a process cell, which provides an overview of process cell utilization
- b) As a list or chart of resource utilizations, which provides a resource or equipment schedule

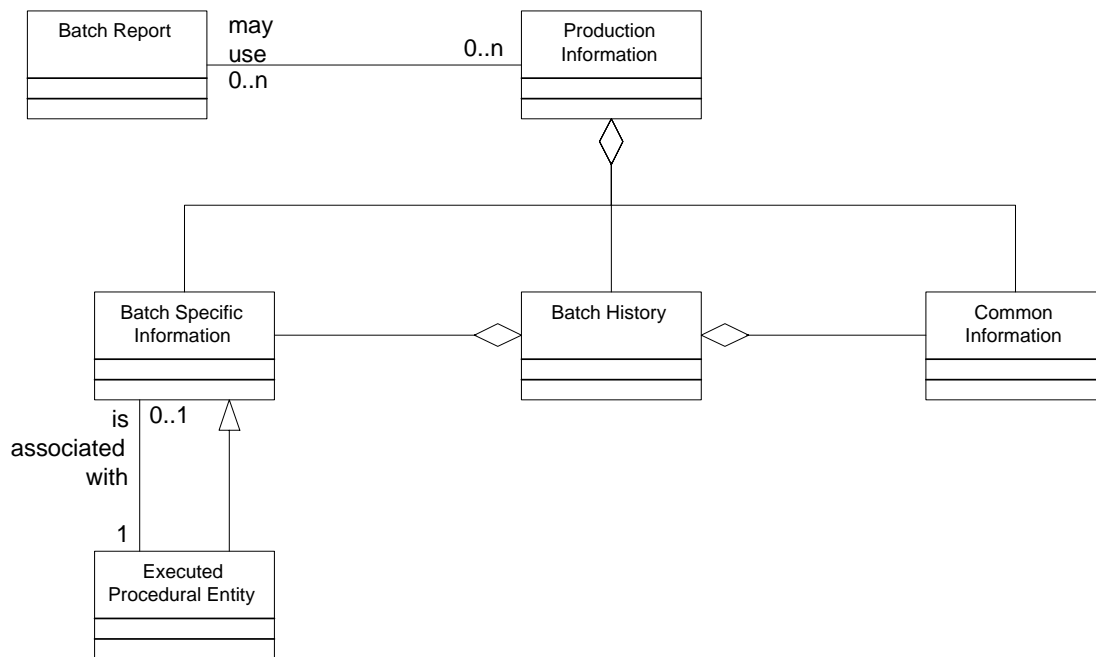
Table 22 — Schedule relation

NAME	SCHEDULE RELATION
FunctionalDescription	A representation of relations between schedule entries (e.g., required line clearance, cleaning between scheduled items, specifications of sequential relations within a procedure).
ATTRIBUTES	
ExecuteOrder	Specifies in what sequence multiple schedule entries should be processed relative to each other.

4.6 Production information management

This subclause describes models that define the collection of production information.

Production information, including timely information of how things have progressed, may include both batch-specific information and selected non-batch-specific, or common, information (see figure 12 and tables 23 through 28).

**Figure 12 — Production information****Table 23 — Production information**

NAME	PRODUCTION INFORMATION
FunctionalDescription	Information generated during the production of a batch.
ATTRIBUTES	

Table 24 — Batch specific information

NAME	BATCH SPECIFIC INFORMATION
FunctionalDescription	Data that relates to one batch history entry.
ATTRIBUTES	
BatchID	Defines the actual Batch ID.
EntryID	Provides unique identification.
NewValue	Indicates the current actual value.
EngineeringUnits	Defines the engineering units, if any, that are appropriate for the NewValue.
EquipmentID	Identifies an equipment element that may be associated with the entry.
UTC	Identifies the Universal Coordinated Time (UTC) and date of the recorded entry.
UserID	Identifies the user, if any, who is associated with the change.

Table 25 — Batch history

NAME	BATCH HISTORY
FunctionalDescription	An item of information that documents batch production.
ATTRIBUTES	
BatchID	Defines the actual Batch ID.

Table 26 — Common information

NAME	COMMON INFORMATION
FunctionalDescription	Data that relates to more than one batch history entry (e.g., cooling water temperature, atmospheric pressure, steam capacity).
ATTRIBUTES	
EntryID	Provides unique Identification.
NewValue	Indicates the current actual value.
EngineeringUnits	Defines the engineering units, if any, that are appropriate for the NewValue.
EquipmentID	Identifies an equipment element that may be associated with the entry.
UTC	Identifies the Universal Coordinated Time (UTC) and date of the recorded entry.
UserID	Identifies the user, if any, who is associated with the change.

Production information may include some of the following:

- a) A copy of the control recipe
- b) A copy of the master recipe

- c) Information about materials that are used and produced
- d) Trend information
- e) Alarms and messages
- f) Operator interactions with the batch (e.g., overwrites, acknowledgements)
- g) Late records and asynchronous records (e.g., lab sample measurements)
- h) Additional information (e.g., allocation, start/stop)

An executed procedural entity is a recording of an instance of execution of a recipe entity or an equipment procedural entity (see table 25). Data is associated with the execution instance and that data is maintained in the related batch history records.

The executed procedural entities that result from the execution of a control recipe will frequently have the same structure as the control recipe entities. The structure of executed procedural elements may, however, deviate from the control recipe in some cases. The following are some typical examples:

- a) Repeated instances of a recipe entity created through looping in the procedural logic
- b) Not executed instances due to branches or GoTos in the procedural logic
- c) Recipe entities that are inserted or repeated manually
- d) Recipe entities that are activated on the spot in units that are already associated with a batch or that are manually associated with the batch

The recipe entity relation and the rest of the recipe entity design pattern is not repeated here, because the history is about logging the facts as they were and not the intended structure.

Table 27 — Executed procedural entity

NAME	EXECUTED PROCEDURAL ENTITY
FunctionalDescription	A representation of a recipe entity (e.g., equipment procedural element) that has been executed.
ATTRIBUTES	
ExecutedProceduralEntityID	Provides a unique identification.
ProceduralEntityCounter	Uniquely identifies repeated execution of the same procedural entity.

Batch reports (see table 28) are understood to be any extraction of batch data from production information for display on screen or on paper, or for transfer to other systems.

Table 28 — Batch report

NAME	BATCH REPORT
FunctionalDescription	A component of a batch report.
ATTRIBUTES	
ReportID	Provides unique identification.

5 Relational tables for information exchange

ISA hereby grants a non-exclusive, royalty-free, limited license under ISA's copyright in the standard, to copy, display and distribute this section of this ISA standard (including software included in or defined by such section), as follows:

1. Producers of products or services intended to comply with the standard may incorporate this designated section, but only to the extent reasonably necessary to make, use, and distribute any product or service (including product documentation) that is compliant with this standard.
2. End users of a product or service made by a producer acting under the preceding license may reproduce and use the designated section, but only to the extent reasonably necessary to enjoy the intended functions of the product or service and to maintain, configure, or reconfigure systems to be compliant.
3. Persons providing education on or promotion of the standard may copy, display and distribute the designated section, but only to the extent reasonably necessary to provide information related to the standard.

Except as expressly permitted, all other reproduction and distribution without permission of ISA is prohibited. All copies of this section of the standard made or distributed under this license must cite the standard and include the following notice of copyright:

Copyright © 2001 by ISA–The Instrumentation, Systems, and Automation Society. All rights reserved. Used with permission of ISA.

5.1 Introduction

This clause defines the structure of SQL relational tables for the exchange of selected batch control related information between systems. It defines an interface specification, meeting the requirements in clause 4, for the exchange of batch information in the following selected categories:

- a) Master and control recipe information
- b) Process cell equipment information
- c) Schedule information
- d) Production information

Exchange tables shall be built using the table names, field names and relations that are defined in this clause. Not all tables need to be implemented, but all fields in an implemented table shall be included. Any such implementation shall be consistent with the table definitions presented here and the concepts of Part 1.

The format for the exchange tables only defines standard information that can be exchanged. The table definitions can be extended through the inclusion of additional attributes, additional related tables and additional enumerations. The extended structures may be used to exchange information among tools that understand the structures, but this information is outside the scope of this standard.

Examples of additional information are

- a) the addition of icons that represent different elements of the equipment hierarchy, so that the equipment may be represented consistently by different tools; and
- b) the addition of control system addresses for equipment element procedures, procedure parameters and data elements.

5.1.1 Method

The relational table structures are defined using SQL, as specified in ISO/IEC 9075:1992, *Information processing systems – Database language SQL with integrity enhancement*.

The exchange mechanism is based on a common structure for database tables. These tables are defined as a data base schema that can be defined in SQL tables. Annex B contains the SQL table definitions of the exchange tables.

Figure 13 illustrates how the exchange tables would be used to exchange batch information between different tools. Each tool will generally have its own local data stores for batch information and each tool is responsible for importing from, and exporting to, the exchange tables.

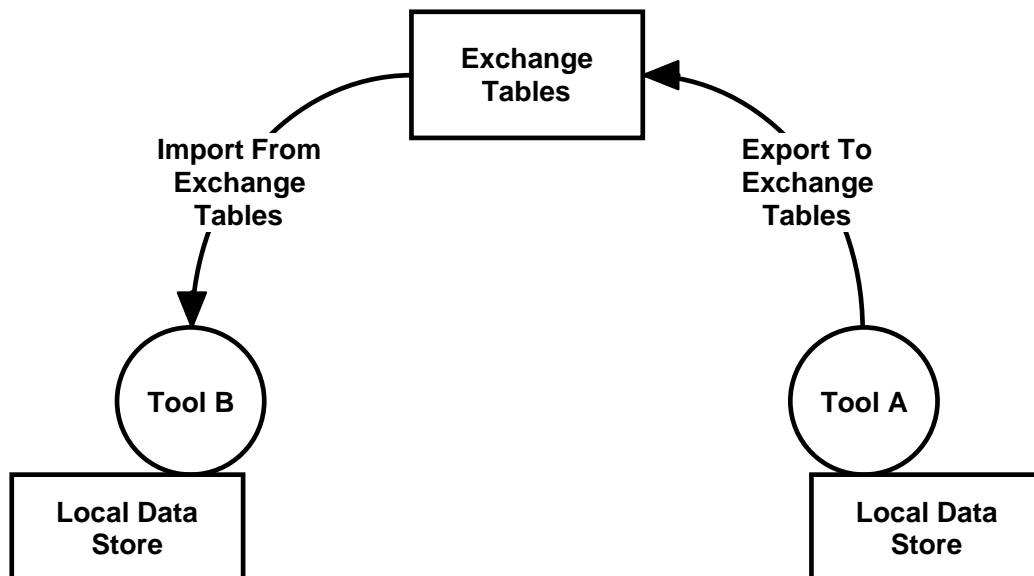


Figure 13 — Data transfer using exchange tables

Most of the remaining models in this clause are represented using entity relationship diagrams (ERDs) (see clause A.3).

The type and use of the tools that are using the batch information are not specified in this standard. Possible tools include, but are not limited to,

- a) recipe authoring systems;
- b) recipe execution systems;
- c) documentation systems;
- d) configuration management systems;
- e) simulation systems;
- f) batch control systems;
- g) planning and scheduling systems; and
- h) information management systems.

5.1.2 Exchange tables

This standard does not specify which tools create and maintain the exchange tables. This standard only defines the structure of the tables. Tools could be designed such that they can only read from pre-existing exchange tables, only write to pre-existing exchange tables, read and write to pre-existing tables, or create, read, and write to exchange tables.

The exchange table structure is designed to allow multiple recipes to be exchanged in the same set of tables and it allows multiple versions of the same recipe to be exchanged. The exchange table structure also allows the exchange of the equipment capability and specification information about process cells and process cell equipment.

The exchange table structure also allows the transfer of either complete or incomplete subsets of a master recipe, equipment descriptions, schedule information, or batch history.

The syntax of data strings in the exchange tables (e.g., computed formula items, transition conditions) is not defined in this standard. Tools that read and write the SQL exchange tables should resolve syntax differences.

The field lengths that are defined in the SQL definition of the exchange tables represent default values and they are not intended to enforce a standard, minimum, or maximum field length. Tools that read and write the SQL exchange tables shall resolve field length differences.

5.1.3 Common exchange information

A set of tables is defined to contain information that describes the exchange format and that can be commonly used in the different exchange table sets. Figure 14 shows the tables that are involved in the common exchange information.

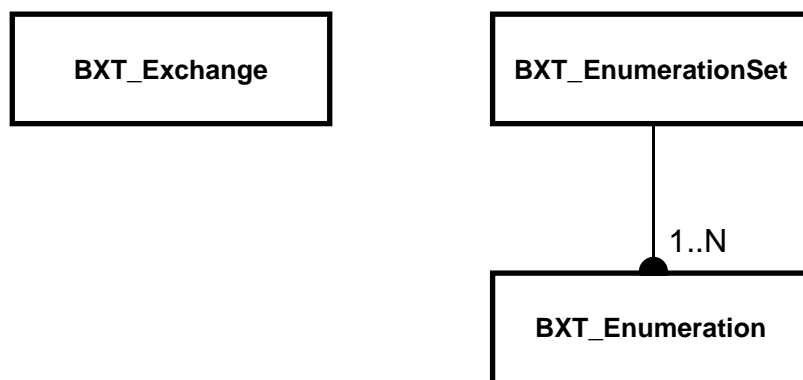


Figure 14 — Common information exchange tables

5.1.3.1 Exchange information

The common exchange information table, BXT_Exchange, contains all of the information that is needed only once in the exchange of data (see table 29).

The table, BXT_Exchange table contents (see table 30), contains one record for each of the defined items (e.g., SCHEMA and the DELIMITER characters). It may also contain other user-defined information.

Table 29 — BXT_Exchange

ATTRIBUTE	DESCRIPTION
<i>ExchangeID</i>	Identifies the exchange element.
ExchangeValue	Identifies the value for the exchanged information.

Table 30 — BXT_Exchange table contents

EXCHANGEID	DESCRIPTION
Schema	The version of the standard that the database schema was defined against. The initial version value shall be the title of this standard (e.g., "ISA-88.00.02-2001").
Delimiter	The character that is used to differentiate between the element names in the recipe element hierarchical name.
ToolID	The identification of the tool that created the exchange tables.
ToolVersion	The version of the tool that created the exchange tables.
ToolSchema	The customized schema version created by the tool.

5.1.3.2 Enumeration sets

Many of the tables contain fields that can contain standard or user-defined enumerated items. These enumerated items are passed as numbers in the exchange tables, with the strings contained in an enumeration set table. The enumeration set tables (see table 31) provides a single location for translation

of the strings between different languages. The BXT_EnumerationSet table defines the enumeration set. The BXT_Enumeration table defines the members of the set and the associated numerical value.

The enumeration tables contain standard enumerations and values. These may be extended by the user. Additional user enumerations of the standard enumeration sets may take on values of 100 and above. Enumeration values of 0-99 are either used or are reserved by this standard. In addition, user defined enumeration sets and their respective values may also be defined in the enumeration tables. For example, the enumeration set "Blend Oil" may be defined for phase parameters with members of "Virgin Oil," "Blended Oil," and "Recovered Oil" having values of 101, 102, and 103, respectively.

Table 31 — BXT_EnumerationSet

ATTRIBUTE	DESCRIPTION
EnumSet	Identifies the standard enumeration set.
Description	Contains the use of the enumeration set. (Provided to assist in the translation of the TextString.)

Table 32 defines the standard enumeration sets that are defined in this standard.

Table 32 — Standard enumeration sets

ENUMSET	DESCRIPTION
Boolean	Defines a set of Boolean values.
DirectionType	Defines how a parameter is intended to be handled.
EquipmentLevel	Defines the equipment hierarchical level for equipment elements.
EquipmentType	Defines the type of equipment record for equipment elements.
EvaluationRule	Defines the evaluation rules for equipment properties.
FormulaType	Defines the recipe formula types.
FormulaSubType	Specifies user-supplied formula sub type definitions.
LinkDepiction	Defines how links between recipe elements are to be depicted.
LinkToType	Defines if a link is referencing a step or transition.
LinkType	Defines the type of link.
RE_Type	Defines the recipe element (RE), either recipe procedure level or allocation symbol.
RE_Use	Defines how a recipe element (RE) is used in a recipe.
RecipeStatus	Definition of the possible status of a recipe.
RecordSet	Defines the enumeration set that is used to classify a record into a category of batch history information.
RecordSetControlRecipe	Provides further history record classification under the category of ControlRecipe.
RecordSetMasterRecipe	Provides further history record classification under the category of MasterRecipe.
RecordSetExecutionInfo	Provides further history record classification under the category of ExecutionInfo.
RecordSetMaterialInfo	Provides further history record classification under the category of MaterialInfo.
RecordSetContinuousData	Provides further history record classification under the category of ContinuousData.
RecordSetEvents	Provides further history record classification under the category of Events.
RecordSetOperatorChange	Provides further history record classification under the category of OperatorChange.
RecordSetOperatorComment	Provides further history record classification under the category of OperatorComment.
RecordSetAnalysisData	Provides further history record classification under the category of AnalysisData.
RecordSetLateRecord	Provides further history record classification under the category of LateRecord.
RecordSetRecipeData	Provides further history record classification under the category of RecipeData.
RecordSetRecipeSpecified	Provides further history record classification under the category of RecipeSpecified.
RecordSetSummaryData	Provides further history record classification under the category of SummaryData.
ScheduleAction	Defines the intended action of the schedule record.
ScheduleMode	Defines the mode in which the schedule record begins execution.
ScheduleStatus	Defines the possible status of a schedule.
SE_Type	Defines the type of entity in a schedule record.
ValueDataType	Defines the data type of an associated data value.
ValueType	Defines how a value string is interpreted.

Table 33 shows how enumerations are defined by this standard.

Table 33 — BXT_Enumeration

ATTRIBUTE	DESCRIPTION
EnumSet	Identifies the name of the enumeration set.
EnumValue	Specifies the numerical value associated with the enumeration member.
EnumString	Defines the associated text for the enumeration member.
Description	Contains the use of the enumeration member. (Provided to assist in the translation of the TextString.)

Table 34 contains the list of standard enumeration members that are defined by this standard.

Table 34 — Standard enumerations

ENUMSET	ENUM VALUE	ENUMSTRING	DESCRIPTION
Boolean	0	FALSE	Definition of a Boolean value.
	1	TRUE	
DirectionType	0	Invalid	Entry not valid
	1	Internal	Identifies how a parameter is handled. Internal = only available within the Recipe Element. Defined at creation or created as an intermediate value.
	2	Input	The Recipe Element receives the value from an external source.
	3	Output	The Recipe Element creates the value and makes it available for external use.
	4	Input/Output	The Recipe Element and external element exchange the value, and may change its value.
	5-99		Reserved
	100+		User defined
EquipmentLevel	0	Invalid	Entry not valid
	1	Enterprise	Identifies the equipment hierarchical level for BXT_EquipElement.
	2	Site	
	3	Area	
	4	Process Cell	
	5	Unit	
	6	Equipment Module	
	7	Control Module	
	8-99		Reserved
100+		User defined	
EquipmentType	0	Invalid	Entry not valid
	1	Class	Identifies the record type for BXT_EquipElement
	2	Element	
	3-99		Reserved
	100+		User defined

Table 34 — Standard enumerations (continued)

ENUMSET	ENUM VALUE	ENUMSTRING	DESCRIPTION
EvaluationRule	0	Invalid	Entry not valid
	1	=	Equals comparison operator for equipment properties.
	2	<>	Not equals comparison operator for equipment properties.
	3	<	Less than comparison operator for equipment properties.
	4	>	Greater than comparison operator for equipment properties.
	5	<=	Less than or equals comparison operator for equipment properties.
	6	>=	Greater than or equals comparison operator for equipment properties.
	7	Member	Is a member of comparison operator for equipment properties.
	8	Notmember	Is not a member of comparison operator for equipment properties.
	9	Not	Not comparison operator for equipment properties.
	10-99		Reserved
100+		User defined	
FormulaSubType	0	Invalid	Entry not valid
	1-99		Reserved
	100+		User defined. Allows further user classification of a formula type.
FormulaType	0	Invalid	Entry not valid
	1	Process Input	Recipe formula type
	2	Process Output	
	3	Process Parameter	
	4-99		Reserved
	100+		User defined
LinkDepiction	0	Invalid	Entry not valid
	1	None	No link depiction
	2	Line	Link shown with line only.
	3	ID	Link shown with identifier only.
	4	Line & ID	Link shown with line and identification.
	5	Line & Arrow	Link shown with line and material flow arrow.
	6	Line, Arrow, & ID	Link shown with line, material flow arrow and identification.
	7-99		Reserved
	100+		User defined

Table 34 — Standard enumerations (continued)

ENUMSET	ENUM VALUE	ENUMSTRING	DESCRIPTION
LinkToType	0	Invalid	Entry not valid
	1	Recipe Element	Link is referencing a record in the BXT_MRecipeElement table.
	2	Transition	Link is referencing a record in the BXT_MRecipeTransition table.
	3-99		Reserved
	100+		User defined
LinkType	0	Invalid	Entry not valid
	1	ControlLink	Defines a link between recipe elements that indicates a flow of procedural control.
	2	TransferLink	Defines a link between recipe elements that indicates a material transfer.
	3	SynchronizationLink	Defines a link between recipe elements where there is some form of synchronization.
	4-99		Reserved
	100+		User defined
RE_Type	0	Invalid	Entry not valid
	1	Master Recipe	Specifies the type of recipe element.
	2	Procedure	
	3	Unit Procedure	
	4	Operation	
	5	Phase	
	6	Allocation	
	7	Begin	
	8	End	
	9	Start Parallel	
	10	End Parallel	
	11	Start Branch	
	12	End Branch	
	13-99		Reserved
100+		User defined	

Table 34 — Standard enumerations (continued)

ENUMSET	ENUM VALUE	ENUMSTRING	DESCRIPTION
RE_Use	0	Invalid	Entry not valid
	1	Linked	A recipe element (RE) may have several referencing RE Steps.
	2	Embedded	A RE has only one referencing RE; one RE is defined for each use of the RE.
	3	Copied	The same as Embedded, but the specific RE was modified from its original definition.
	4-99		Reserved
	100+		User defined
RecipeStatus	0	Invalid	Entry not valid
	1	Approved for Production	Recipe is approved for production.
	2	Approved for Test	Recipe is only approved for test.
	3	Not Approved	Recipe is not approved for production or test.
	4	Inactive	Recipe is not active.
	5	Obsolete	Recipe is obsolete.
	6-99		Reserved
	100+		User defined
RecordSet	0	Invalid	Entry not valid
	1	RecordSetControlRecipe	Defines that a batch history information record is part of the ControlRecipe category.
	2	RecordSetMasterRecipe	
	3	RecordSetExecutionInfo	
	4	RecordSetMaterialInfo	
	5	RecordSetContinuousData	
	6	RecordSetEvents	
	7	RecordSetOperatorChange	
	8	RecordSetOperatorComment	
	9	RecordSetAnalysisData	
	10	RecordSetLateRecord	
	11	RecordSetRecipeData	
	12	RecordSetRecipeSpecified	
	13	RecordSetSummaryData	
	14-99		Reserved
100+		User defined	

Table 34 — Standard enumerations (continued)

ENUMSET	ENUM VALUE	ENUMSTRING	DESCRIPTION
RecordSet ControlRecipe	0	Invalid	Entry not valid
	1	Entire Control Recipe	History record is related to the entire control recipe.
	2-99		Reserved
	100+		User defined
RecordSet MasterRecipe	0	Invalid	Entry not valid
	1	Entire Master Recipe	History record is related to the entire master recipe.
	2-99		Reserved
	100+		User defined
RecordSet ExecutionInfo	0	Invalid	Entry not valid
	1	Allocation	
	2	De-allocation	
	3	State Change	
	4	State Command	
	5	Mode Change	
	6	Mode Command	
	7	Procedural Entity Message	
	8	Procedural Entity Alarm	
	9	Procedural Entity Version	
	10	Procedural Entity Prompt	
	11	Procedural Entity Prompt Resp	
	12-99		Reserved
100+		User defined	
RecordSet MaterialInfo	0	Invalid	Entry not valid
	1	Material Consumption	
	2	Material Production	
	3	Material Allocation	
	4	Material De-allocation	
	5-99		Reserved
	100+		User defined

Table 34 — Standard enumerations (continued)

ENUMSET	ENUM VALUE	ENUMSTRING	DESCRIPTION
RecordSet ContinuousData	0	Invalid	Entry not valid
	1	Continuous Data Value	
	2	Trend Association	
	3	Trend Disassociation	
	4-99		Reserved
	100+		User defined
RecordSet Events	0	Invalid	Entry not valid
	1	General Event	
	2-99		Reserved
	100+		User defined
RecordSet OperatorChange	0	Invalid	Entry not valid
	1	General Operator Intervention	
	2-99		Reserved
	100+		User defined
RecordSet OperatorComment	0	Invalid	Entry not valid
	1	General Operator Comment	
	2-99		Reserved
	100+		User defined
RecordSetAnalysis Data	0	Invalid	Entry not valid
	1	General Analysis Message	
	2-99		Reserved
	100+		User defined
RecordSet LateRecord	0	Invalid	Entry not valid
	1	General Late Record	
	2-99		Reserved
	100+		User defined

Table 34 — Standard enumerations (continued)

ENUMSET	ENUM VALUE	ENUMSTRING	DESCRIPTION
RecordSetRecipe Data	0	Invalid	Entry not valid
	1	Generic Recipe Data	
	2	Recipe Parameter Value Change	
	3	Recipe Result Data	
	4-99		Reserved
	100+		User defined
RecordSet RecipeSpecified	0	Invalid	Entry not valid
	1	Generic Recipe Specified Data	
	2-99		Reserved
	100+		User defined
RecordSet SummaryData	0	Invalid	Entry not valid
	1	Generic Summary Data	
	2	Utilities Consumption	
	3	Equipment Run Time	
	4-99		Reserved
	100+		User defined
ScheduleAction	0	Invalid	Entry not valid
	1	New	Schedule record change action
	2	Update	
	3	Delete	
	4-99		Reserved
	100+		User defined
ScheduleMode	0	Invalid	Entry not valid
	1	Automatic	Schedule record mode
	2	Semi-automatic	
	3	Manual	
	4	Not Specified	
	5-99		Reserved
	100+		User defined

Table 34 — Standard enumerations (continued)

ENUMSET	ENUM VALUE	ENUMSTRING	DESCRIPTION
ScheduleStatus	0	Invalid	Entry not valid
	1	Complete	Batch schedule record status
	2	In-progress	
	3	Scheduled	
	4	Schedule Hold	
	5	Not Specified	
	6-99		Reserved
	100+		User defined
SE_Type	0	Invalid	Entry not valid
	1	Campaign	Defines the type of Scheduled Entry.
	2	Batch	
	3	Unit procedure	
	4	Operation	
	5	Phase	
	6-99		Reserved
	100+		User defined
ValueDataType	0	Invalid	Entry not valid
	1	Boolean	Defines the data type that is expected for an associated value.
	2	8-Bit string	
	3	16-Bit string	
	4	32-Bit string	
	5	8-Bit unsigned integer	
	6	16-Bit unsigned integer	
	7	32-Bit unsigned integer	
	8	8-Bit signed integer	
	9	16-Bit signed integer	
	10	32-Bit signed integer	
	11	32-Bit float	
	12	Double float	
	13	Octet string	
	14	DateTime	
	15-99		Reserved
	100+		User defined

Table 34 — Standard enumerations (continued)

ENUMSET	ENUM VALUE	ENUMSTRING	DESCRIPTION
ValueType	0	Invalid	Entry not valid
	1	Constant	Defines how a value string is interpreted. It contains a fixed value as a string.
	2	Reference	Defines how a value string is interpreted. It points to the source of the value.
	3	Equation	Defines that a value string represents an expression to be evaluated in order to determine the value.
	4	External	Value is supplied by some external means, and it is not contained in the recipe (i.e., value may be supplied by an operator or by a scheduling system).
	5-99		Reserved
	100+		User defined

5.2 Master recipe information

This subclause only deals with *Master Recipes*. The information that is exchanged is what is needed in order to exchange a master recipe, as defined in Part 1. This information contains the procedural control definitions, the formula value definitions, the equipment requirements of the recipe, header information, process cell specific information, other information, and coordination control requirements.

The information that is exchanged is what is needed to **exchange** a master recipe, but this information does not specify the following:

- a) How the information was created
- b) How the master recipe could be used in a system

5.2.1 Recipe definitions

Creation of a master recipe, as specified in Part 1, may use information from the site recipe and the definition of the process cell's processing capability. A master recipe is tied to the process cell's processing capability, but the definition of the processing capability is not part of the recipe-exchanged information.

5.2.2 Recipe structure

A recipe is made up of the following information categories: header, formula, procedure, equipment requirements, and other information.

The recipe exchange schema is recursive by nature. The basic structure of that schema, as shown in figure 15, is built around these five information categories. Every level of the definition contains all of these information categories until the procedural definition references an equipment procedural entity.

The term Recipe Element (RE) is used to define some of the structural entities in a recipe. A recipe element may be the Master Recipe itself, Recipe Procedure, Recipe Unit Procedure, Recipe Operation, Recipe Phase, Allocation symbol, or other graphical symbols, as defined in table 34.

In the recursive model of the schema, recipe elements contain either lower-level recipe elements or references to Equipment Procedural Elements. While the definition of the recipe element is contained in the recipe element table, each use of a recipe element is called a Step.

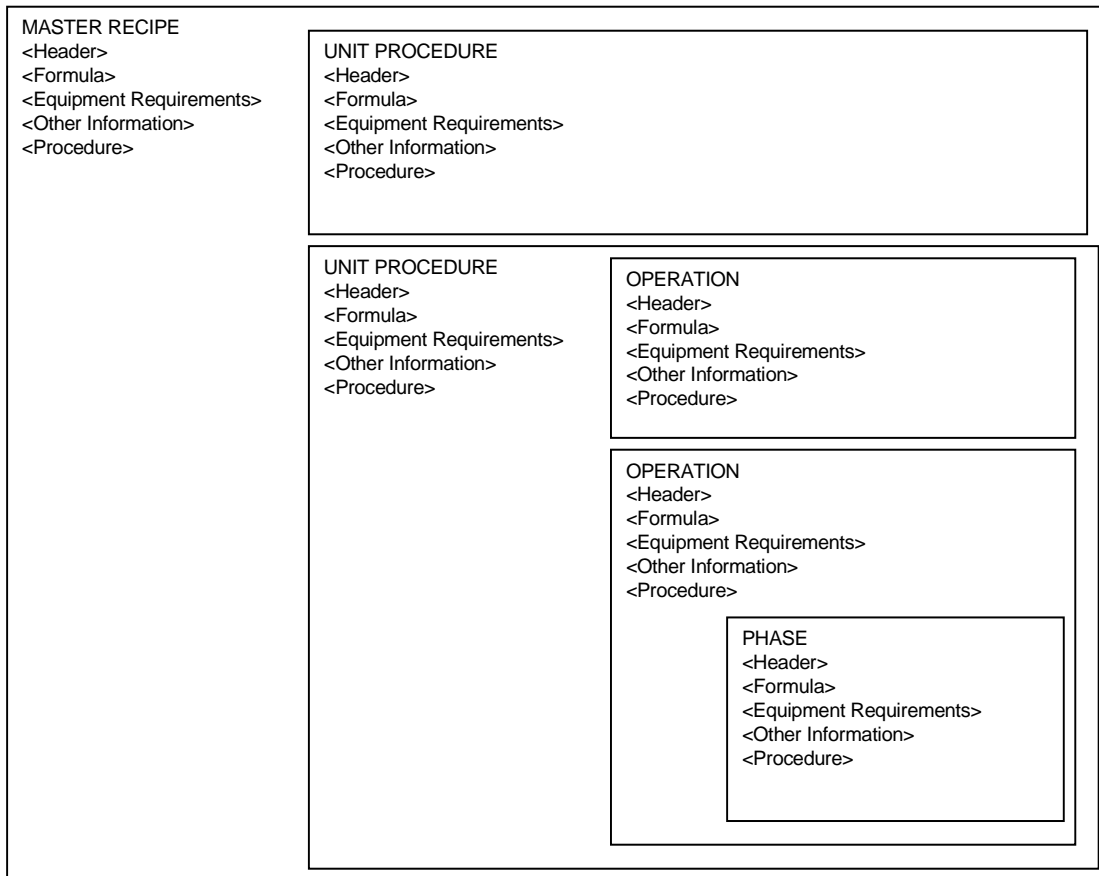


Figure 15 — Nested recipe elements make up a recipe

5.2.3 Table overview and integrity constraints

Figure 16 illustrates the tables that are used to exchange recipes and their relationships and it defines the associated integrity constraints between the tables.

Non-direct relationships, such as between the LINK and RE, are not depicted; however, they are defined through the set of common key fields in the table. "NOT NULL" entries in the associated SQL tables are used only to enforce the integrity constraints of the Entity Relationship diagram.

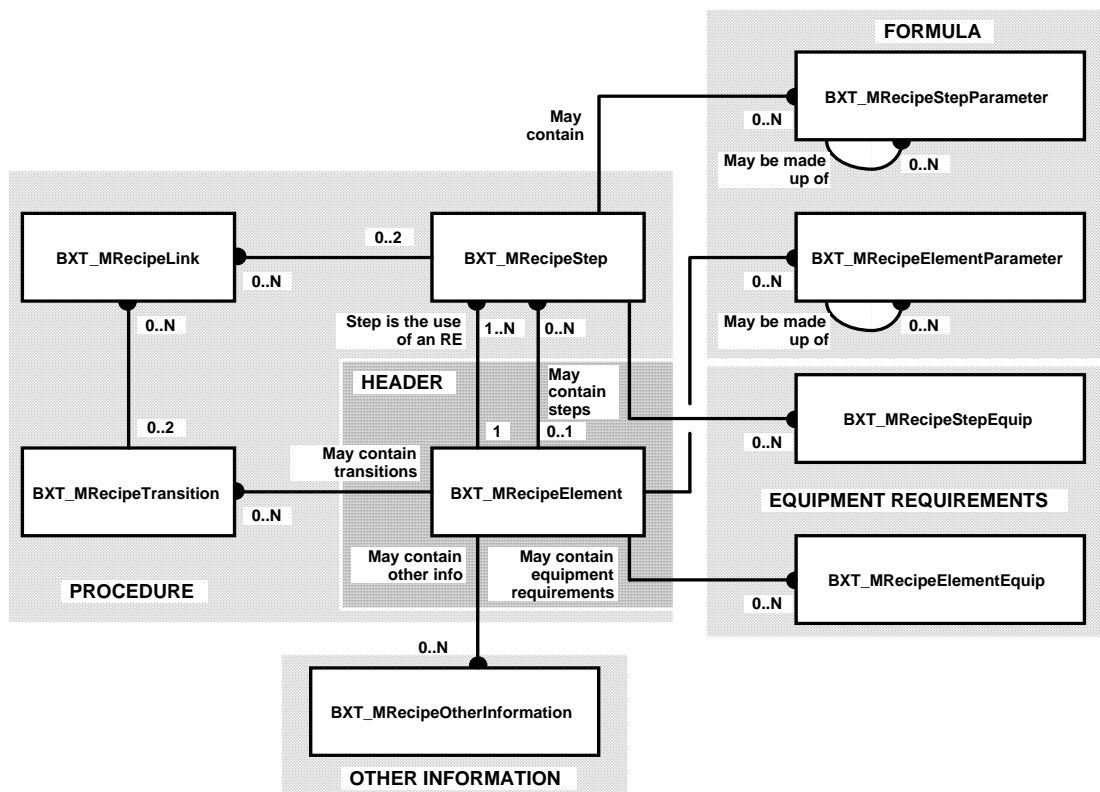


Figure 16 — Exchange table relationships

The recursive definition of the Recipe Elements (REs) is through the two associations between the BXT_MRecipeStep and BXT_MRecipeElement entities. Each defined use of a recipe element is recorded in the BXT_MRecipeStep table. Each definition of a recipe element is recorded in the BXT_MRecipeElement table. One association shows what steps are contained in a recipe element. The other association shows what element is referenced by a step. Each RE that is referenced by the owning RE has a record in the Step table. The Step table then references the actual RE definition. The tables support a single RE definition per Step, and they support multiple Steps that reference the same RE.

The relationship between Steps and Transitions is maintained in the BXT_MRecipeLink table.

The master recipe formula definition is the collection of all master recipe parameter records and it describes the process input, process output, and process parameters within the recipe. The formula values are contained in the BXT_MRecipeStepParameter table and they have their definition in the BXT_MRecipeElementParameter table.

The recipe's equipment requirements are contained in the BXT_MRecipeStepEquip and BXT_MRecipeElementEquip tables.

Figure 17 shows how entries in each table are related to each other for the BXT_MRecipeStep, BXT_MRecipeElement, BXT_MRecipeElementParameter, and BXT_MRecipeStepParameter tables. One BXT_MRecipeElement record exists for each version of an exchanged recipe. A relationship between this BXT_MRecipeElement record and a single BXT_MRecipeStep table record exists, and this relationship

contains the specific recipe information, including formula values in the BXT_MRecipeStepParameter table.

The BXT_MRecipeStep table contains a key into a single record in the BXT_MRecipeElement table. This BXT_MRecipeElement table record contains the definition of the recipe's procedure, including the definition of the formula values in the BXT_MRecipeElementParameter table.

The BXT_MRecipeElement table for this procedure contains a key into multiple records in the BXT_MRecipeStep table, one for each Unit Procedure. (Other procedure level recipe elements are omitted from this example for simplicity.) The BXT_MRecipeStep record for the Unit Procedures contains a key to the BXT_MRecipeElement record that defines the Unit Procedure. The BXT_MRecipeElement record for each Unit Procedure contains a key into the BXT_MRecipeStep table for each Operation. This structure continues down to the Phase definitions.

This table format uses the BXT_MRecipeStep and BXT_MRecipeElement tables to contain the procedural hierarchy of a recipe's procedure.

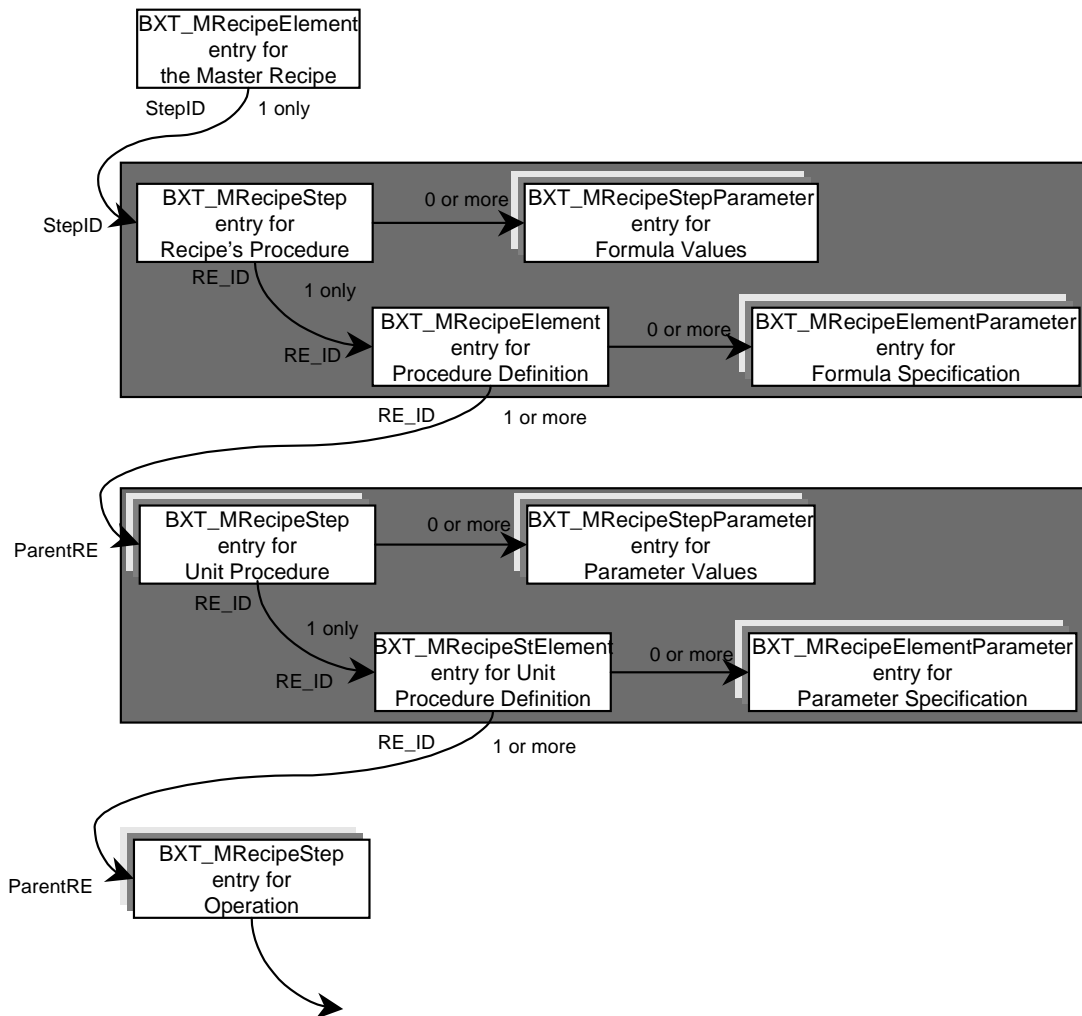


Figure 17 — How entries relate in the tables

5.2.4 Recipe table summary

The tables that are defined for recipe exchange are shown in table 35.

Table 35 — Recipe exchange tables

TABLE NAME	DESCRIPTION
BXT_MRecipeStep	One record for each use of a RE within a RE.
BXT_MRecipeElement	One record for each recipe element that is exchanged.
BXT_MRecipeTransition	One record for each transition that is used within a RE.
BXT_MRecipeLink	One record for each link between Steps and Transitions.
BXT_MRecipeElementParameter	One record for each parameter for each RE.
BXT_MRecipeStepParameter	One record for each parameter for each Step.
BXT_MRecipeOtherInformation	One record for each element of other information.
BXT_MRecipeElementEquip	One record for each property requirement for a RE.
BXT_MRecipeStepEquip	One record for each value for an equipment property defined in a step.

5.2.5 Recipe table definitions

5.2.5.1 Header information

Master recipe header information is transferred as fields of a record in the BXT_MRecipeElement table. The BXT_MRecipeElement table (see table 36) contains one element for each master recipe that will be exchanged. The combination of RE_ID and REVersion defines the exchanged recipe master recipe.

Table 36 — BXT_MRecipeElement

ATTRIBUTE	DESCRIPTION
<i>RE_ID</i>	Identifies the Recipe Element that will be exchanged (e.g., <i>Red Oak</i>). When combined with the "version," this field defines a unique instance of a RE. When the record represents the master recipe, this field contains the master recipe ID.
<i>REVersion</i>	Identifies the version of the RE. When combined with a "RE_ID," this field defines a unique instance of a RE (e.g., V10.3).
VersionDate	Identifies the date and time that this version of the RE was last modified.
ApprovalDate	Identifies the date and time that this version of the recipe was approved.
EffectiveDate	Identifies the date and time that this version of the recipe is effective.
ExpirationDate	Identifies the date and time that this version of the recipe expires.
Author	Identifies the person or system that authored this version (e.g., J. Smith).

Table 36 — BXT_MRecipeElement (continued)

ATTRIBUTE	DESCRIPTION
ApprovedBy	Identifies the person or system that approved this version of the recipe.
ProcessCellID	Identifies the process cell or class of process cells for which this version of the master recipe was defined.
ProductID	Identifies the product or product family that would be created by execution of this version of the recipe (e.g., Premium Beer).
UsageConstraint	Defines other rules that determine the usage (e.g., “must be succeeded by...” or “must not run in parallel with...”).
Description	Describes the recipe element.
Status	Defines the status of the information that is being exchanged as an enumeration from the enumeration set "Recipe Status."
RE_Type	Identifies the type of the RE from the enumeration set "RE_Type."
RE_Function	Contains an optional reference to the equipment information exchange tables. The format for this information is not defined in this clause. An example is a reference to an equipment procedural element (see BXT_EquipInterface table). If this entry is NULL, then the function of this RE is defined by BXT_MRecipeStep and BXT_MRecipeTransition entries that have a ParentRE that matches this table's RE_ID.
RE_Use	Identifies the relationship between the RE and the BXT_MRecipeStep, from the enumeration "RE Use." Linked specifies that there may be multiple BXT_MRecipeStep uses of the RE definition. Linked would be used when the BXT_MRecipeElement is a library building block. Embedded specifies that the RE has only one referencing RE and one RE is defined for each use of the RE. In this case, the RE is "embedded" in the recipe definition of a single recipe step and it is not used elsewhere. Copied specifies the same as Embedded, but the RE was modified from some original definition. Copied would be used when the RE was a library building block that was fully reproduced in the recipe, and its linkage to the library removed.
DerivedRE	Identifies the recipe element from which this recipe element was derived.
DerivedVersion	Identifies the version of the recipe element from which this recipe element was derived.

5.2.5.2 Procedure information

The procedural parts of a master recipe are included or contained in a combination or collection of tables. These records define

- a) the procedural control element steps;
- b) the procedural control elements;
- c) allocation symbols and other graphical representation symbols;
- d) the parameterization of procedural elements with limits;
- e) the linkage between the elements; and
- f) the transition definitions between elements.

The Step, Transition, and Link tables contain the definition of REs that contain lower-level REs. The BXT_MRecipeStep table (see table 37) contains each instance of use of a RE. Each BXT_MRecipeStep contains the parameters used when a RE is used. Because a RE may be used multiple times in a recipe, there is only one record in the BXT_MRecipeElement table, but there are multiple records in the BXT_MRecipeStep table, one record for each use of the RE.

Table 37 — BXT_MRecipeStep

ATTRIBUTE	DESCRIPTION
<i>ParentRE</i>	Identifies the RE or master recipe with which this Step is associated.
<i>ParentVersion</i>	Identifies the version of the RE or master recipe with which this Step is associated. When combined with a "ParentRE," this field defines a unique instance of an RE.
<i>StepID</i>	Identifies the unique execution instance of the referenced RE, with a name that is unique to the scope of its parent RE. (A simple example might just be the step number in the RE.)
RE_ID	Identifies the name of the RE that this step is an instance of, with a name that is unique to the scope of its parent RE.
REVersion	Identifies the version of the RE this step is an instance of.
VerticalStart	Specifies the vertical starting position in the presentation of this element in the procedural view of the parent RE, in normalized coordinates of (0,0) upper left to (1,1) lower right.
VerticalStop	Specifies the vertical stopping position in the presentation of this element in the procedural view of the parent RE, in normalized coordinates of (0,0) upper left to (1,1) lower right.
HorizontalStart	Specifies the horizontal starting position in the presentation of this element in the procedural view of the parent RE, in normalized coordinates of (0,0) upper left to (1,1) lower right.
HorizontalStop	Specifies the horizontal stopping position in the presentation of this element in the procedural view of the parent RE, in normalized coordinates of (0,0) upper left to (1,1) lower right.
ScaleReference	Specifies the reference size for recipe elements; all formula values are based on this reference size (e.g., 1234.5 kg).
ScaleEngrUnits	Specifies the units of ScaleReference.
MaximumScale	Specifies the maximum scale factor, or size, of the recipe element.
MinimumScale	Specifies the minimum scale factor, or size, of the recipe element.

5.2.5.2.1 Recipe element

The BXT_MRecipeElement table (see table 36) contains one record for each Procedural Element that is referenced in the exchanged master recipe. This table contains the definition of the Element, not the use of the element. In the table, one record exists for the procedure, for each unit procedure, for each operation, and for each recipe phase that is exchanged. The BXT_MRecipeElement and BXT_MRecipeElementParameter tables contain the specifications for the use of the RE, the number and types of parameters that are passed, and the default values for the parameters.

REs shall be unique to a parent RE. The RE_ID is a fully qualified name of the RE under its parent REs; therefore, the RE_ID is enough to contain all of the parent's RE_IDs.

5.2.5.2.2 Transitions

The BXT_MRecipeTransition table contains one record for each transition connection that is defined by the REs (see table 38). These records correspond to the transitions in Procedure Function Charts (see clause 6).

Table 38 — BXT_MRecipeTransition

ATTRIBUTE	DESCRIPTION
<i>RE_ID</i>	Identifies the RE the transition is contained in.
<i>REVersion</i>	Identifies the version of the RE. When combined with a "RE_ID" defines a unique instance of a master recipe.
<i>TransitionID</i>	Identifies the unique execution instance of this transition element. The ID contains the full hierarchy of parent RE instance names of which this transition is a member element.
Condition	Contains the expression or condition that, if TRUE, causes or allows the transition.
VerticalStart	Specifies the vertical starting position in the presentation of this element in the procedural view of the RE, in normalized coordinates of (0,0) upper left to (1,1) lower right.
VerticalStop	Specifies the vertical stopping position in the presentation of this element in the procedural view of the RE, in normalized coordinates of (0,0) upper left to (1,1) lower right.
HorizontalStart	Specifies the horizontal starting position in the presentation of this element in the procedural view of the RE, in normalized coordinates of (0,0) upper left to (1,1) lower right.
HorizontalStop	Specifies the horizontal stopping position in the presentation of this element in the procedural view of the RE, in normalized coordinates of (0,0) upper left to (1,1) lower right.

5.2.5.2.3 Links

The BXT_MRecipeLink table contains one record for each connection that is defined in the REs and/or transitions (see table 39). These records correspond to the lines that connect elements in the Procedure Function Charts that are described in clause 6.

Table 39 — BXT_MRecipeLink

ATTRIBUTE	DESCRIPTION
<i>RE_ID</i>	Identifies the RE with which the Step and/or Transition is associated.
<i>REVersion</i>	Identifies the version of the RE. When combined with a "RE_ID," this field defines a unique instance of a RE.
<i>LinkID</i>	Specifies a unique ID for the link, which simplifies access to the table.
FromType	Kept as an enumeration, defines if the FromElement specifies a StepID for a step or a TransitionID for a transition, from the enumeration set "LinkToType."
FromElement	Specifies the Step name or TransitionID. The ID contains the full hierarchy of parent RE instance names of which this element is contained. This shall match the step or transition name in the step or transition table.
ToType	Kept as an enumeration, defines if the ToElement specifies a StepID for a step or a TransitionID for a transition, from the enumeration set "LinkToType."
ToElement	Specifies the Step name or TransitionID. The ID contains the full hierarchy of parent RE instance names of which this element is contained. This shall match the step or transition name in the step or transition table.
LinkType	Specifies if the link is a procedural control flow or a material transfer association. Kept as an enumeration, as ControlLink or TransferLink, from the enumeration set "LinkType."
VerticalStart	Specifies the vertical starting position in the presentation of this element in the procedural view of the parent RE, in normalized coordinates of (0,0) upper left to (1,1) lower right.
VerticalStop	Specifies the vertical stopping position in the presentation of this element in the procedural view of the parent RE, in normalized coordinates of (0,0) upper left to (1,1) lower right.
HorizontalStart	Specifies the horizontal starting position in the presentation of this element in the procedural view of the parent RE, in normalized coordinates of (0,0) upper left to (1,1) lower right.
HorizontalStop	Specifies the horizontal stopping position in the presentation of this element in the procedural view of the parent RE, in normalized coordinates of (0,0) upper left to (1,1) lower right.
Depiction	Defines how the link is presented, from the enumeration set "LinkDepiction."
EvaluationOrder	Defines the specified order of evaluation of the link (if required) to meet the left-to-right evaluation of PFC transition checks that are specified in clause 6. All links from the same step to multiple transitions are assumed to be evaluated in the order that is specified by the order field. Lower numbers are evaluated first.

5.2.5.2.4 Parameters

The BXT_MRecipeElementParameter table contains one record for each parameter for each RE that is defined (see table 40). For example, a RE called *CHARGE* may be defined with two parameters: the type of material to charge and the amount to charge. One record would exist in the BXT_MRecipeElement table and two records would exist in the BXT_MRecipeElementParameter table for the RE *CHARGE*.

Table 40 — BXT_MRecipeElementParameter

ATTRIBUTE	DESCRIPTION
<i>RE_ID</i>	Identifies the RE with which the parameter is associated.
<i>REVersion</i>	Identifies the version of the RE. When combined with a "RE_ID," this field defines a unique instance of a master recipe.
<i>ParameterID</i>	Identifies the Procedural Element parameter. Note that if the parameter is part of a set, the ParentParamID field is used and the parameter set becomes part of the name of the parameter. For example, for a parameter set of MINOR_CHARGES and a parameter of BLUE_DYE, the ParameterID would be MINOR_CHARGES.BLUE_DYE if the delimiter character is a period.
ParentParamID	Identifies the parent parameter set of which this parameter is a member. This field will be NULL if there is no parameter set.
DataInterpretation	Defines how the default parameter value is interpreted, as an enumeration (i.e., constant, reference, or equation), from the enumeration set "ValueType."
DataDirection	Defines how the parameter value is intended to be handled, as an enumeration (i.e., input, output, input/output, neither), from the enumeration set "DirectionType."
DefaultValue	Contains the default parameter value that is used if the instance of execution does not specify a value and it may be a member of a user enumeration set.
ValueType	Defines the data type of the value, from the enumeration set "ValueDataTypes."
Description	Describes the parameter or the use of the parameter in the RE.
EngrUnits	Identifies the engineering units of measure for the Value. (e.g., kg, pounds).
EnumSet	Identifies the enumeration that this element is a member of (if not NULL).
DefaultScaling	Specifies the default selection when defined as an instance in the BXT_MRecipeStepParameter table and it is kept as an enumeration from enumeration set "Boolean." If TRUE, the formula value will be scaled when the size of the batch is scaled. If FALSE, the formula value will not be scaled when the size of the batch is scaled. Non-linear scaling may be accomplished through the use of expressions in the formula values.
ParamType	Specifies the default selection when defined as an instance in the BXT_MRecipeStepParameter table. This field identifies the use of the formula value as an enumeration from the enumeration set "FormulaType" (i.e., Process Input, Process Output, or a Process Parameter). The enumeration set is user extensible.
ParamSubType	Specifies the default selection when defined as an instance in the BXT_MRecipeStepParameter table. This field identifies the use of the formula value as an enumeration from the FormulaSubType enumeration set. Elements in this set are all user defined.

5.2.5.2.5 Standard sub-parameters

This standard recognizes a set of sub-parameters that may be related to a parameter value, to further qualify its definition and use. For example, defining and transferring the high and low limits that a parameter is allowed to take may be useful. This type of information is transferred in the exchange tables by defining sub-parameters for the affected parameter. Sub-parameters are defined for a given parameter, by creating a new table record with the affected ParameterID used as the ParentParamID. The set of standard sub-parameters and their defined use is given in table 41. Other sub-parameters may also be defined by the user. This set of sub-parameters should be supported for use with the

BXT_MRecipeElementParameter table, the BXT_MRecipeStepParameter table (see 5.2.5.3.1), the BXT_EquipInterfaceParameter table (see 5.3.5.8), and BXT_ScheduleParameter table (see 5.4.3.4).

Table 41 — Standard sub-parameters

PARAMETERID	DESCRIPTION
<i>HighValueLimit</i>	Specifies the largest value that the associated parameter is allowed to take on.
<i>LowValueLimit</i>	Specifies the smallest value that the associated parameter is allowed to take on.
<i>HighTolerance</i>	Specifies the largest upward deviation from the value of the associated parameter that is allowed.
<i>LowTolerance</i>	Specifies the largest downward deviation from the value of the associated parameter that is allowed.

5.2.5.3 Formula

Values of components that are to be used in the production or execution of a batch are passed as step parameters and their limits. The BXT_MRecipeStepParameter table contains one record for each parameter that is used for each Step that is defined (see table 42). Because REs can evaluate default parameter values, not all parameters of a RE need to be defined in the Step.

Table 42 — BXT_MRecipeStepParameter

ATTRIBUTE	DESCRIPTION
<i>ParentRE</i>	Part of the KEY to the BXT_MRecipeStep table for this parameter.
<i>ParentVersion</i>	Part of the KEY to the BXT_MRecipeStep table for this parameter.
<i>StepID</i>	Identifies the unique execution instance of the RE, with a name unique to the scope of its parent RE.
<i>ParameterID</i>	Identifies the RE parameter.
<i>ParentParameterID</i>	Identifies the parent parameter ID if defined. (Allows for parameter sets.)
<i>ParameterValue</i>	Contains the parameter value (numbers are transferred as ASCII representations of the number). If the parameter type is a user enumeration set, then this may be a member of the set.
<i>DataInterpretation</i>	Defines how the parameter value is interpreted, as an enumeration (constant, reference, external, or equation), from the set "ValueType."
<i>Scaled</i>	Kept as an enumeration, from the enumeration set "Boolean." If TRUE, then the parameter value will be scaled when the batch size is scaled. If FALSE, then the formula value will not be scaled when the batch size is scaled.

5.2.5.4 Other information

Other information typically has to be transferred with the recipe (see table 43). The meaning of the information is not specified in the exchange definition, but it is an agreement between the sender and receiver. This other information is usually extra documentation or descriptive information that may be needed in order to exchange a valid master recipe, but it is not information that is needed in order to execute the recipe. Examples of other information may include compliance documentation, molecular structure diagrams or even pictures of the expected product. The RE Other Information table, named BXT_MRecipeOtherInformation, contains this data.

Because the structure and form of this data are so variable, the recipe exchange language provides a means to reference, or point to, this information. The actual information may be exchanged in the DataValue field, if it is text information, or it may reference other files whose names are in the DataValue fields. Exchange of the other files is outside the scope of this standard.

Table 43 — BXT_MRecipeOtherInformation

ATTRIBUTE	DESCRIPTION
<i>RE_ID</i>	Identifies the RE with which the "other" data is associated.
<i>REVersion</i>	Identifies the version of the RE. When combined with a "RE_ID," this field defines a unique instance of the RE.
<i>StepID</i>	Identifies the unique execution instance of the RE if the other information is associated with the step. If null, then the other information is associated with the RE.
<i>DataID</i>	Defines the identification of a data element with other information.
<i>DataType</i>	Identifies the type of the data value from the enumeration set ValueDataType.
<i>DataValue</i>	Specifies the value of the other data information. This field may be the name of a file that contains the actual data. The importing tool should have access to this file.
Description	Describes the data element type.

5.2.5.5 Equipment requirements

RE equipment requirements contain the equipment constraints and conditions of the recipe, and they are defined using similar tables and relations as those that are used to define the equipment capabilities in 5.3.

The equipment constraints and conditions may be associated with any Recipe Element in the Recipe Element hierarchy. RE equipment requirements define the equipment constraints and conditions that are applicable at that level (e.g., at the Unit Procedure or Operation level).

The equipment requirements are contained in the BXT_MRecipeElementEquip table and the BXT_MRecipeStepEquip table. The BXT_MRecipeElementEquip table contains the definition and default value of the property, and the BXT_MRecipeStepEquip table contains the specific property value that is required for a step.

5.2.5.5.1 RE equipment requirement

A RE may have equipment requirements (e.g., "Reactor name" or "Reactor lining"), and these specify the required properties for the RE to execute. Specific allowed values for the property for a specific step are defined in the BXT_MRecipeStepEquip table.

The equipment requirement may define data elements that the process cell supplies for use in the recipe. These elements could be used in the recipe's transition conditions or elsewhere in expressions. Examples of data elements are "VesselPressure" available from a unit and "SteamPressure" from an equipment module.

Table 44, named BXT_MRecipeElementEquip, defines this set of related equipment requirements.

Table 44 — BXT_MRecipeElementEquip

ATTRIBUTE	DESCRIPTION
<i>RE_ID</i>	Identifies the RE for which the equipment requirements are needed. When combined with the "version," this field defines a unique instance of the RE.
<i>REVersion</i>	Identifies the version of the RE. When combined with a "RE_ID," this field defines a unique instance of a RE.
<i>PropertyID</i>	Defines the equipment property needed for the RE to execute (e.g., "Reactor Class" or "Vessel Pressure Tag").
DefaultValue	Defines the default value for the property, if none is specified in the steps (e.g., "Exothermic Reactor" or "VesPressure").
DataInterpretation	Specifies how the value is interpreted, and it is defined as an enumeration (i.e., constant, reference, external, or equation), from the enumeration set "ValueType."
EvaluationRule	Specifies how the value is compared against the equipment property value, and it is defined as an enumeration, from the enumeration set "EvaluationRule."
EngrUnits	Identifies the engineering units of measure for the value (e.g., kg, pounds).
Description	Describes what the property is and it may include why it is needed for the recipe.

5.2.5.5.2 RE step equipment requirement

A BXT_MRecipeStep may have a specific value for an equipment property (e.g., "Reactor Class" of "Exothermic" or "Reactor Lining" = "Glass"). Table 45, named BXT_MRecipeStepEquip, defines this set of related equipment requirements.

Table 45 — BXT_MRecipeStepEquip

ATTRIBUTE	DESCRIPTION
<i>ParentRE</i>	Part of the KEY to the BXT_MRecipeStep table for this parameter.
<i>ParentVersion</i>	Part of the KEY to the BXT_MRecipeStep table for this parameter.
<i>StepID</i>	Identifies the unique execution instance of the RE, with a name unique to the scope of its parent RE.
<i>PropertyID</i>	Identifies the RE property.
PropertyValue	Contains the property value (numbers are transferred as ASCII representations of the number). If the parameter type is a user enumeration set, then this may be a member of the set.

The "meaning" of the constraints is not important to the language definition; however, it is an agreement on the naming of equipment sets or equipment entities.

5.3 Process cell equipment model exchange

A set of tables is defined that describes the capabilities of the equipment in a process cell. This information reflects the actual capabilities of a process cell. The exchange of process cell capabilities can be useful as an information exchange, even if it is not used in conjunction with recipes.

5.3.1 Equipment description

The process cell capabilities tables are organized around a hierarchy of equipment descriptions. The elements in the hierarchy correspond to the elements in the Part 1 equipment hierarchy.

The equipment information exchange tables include the following information:

- a) Equipment element - This defines a specific equipment element or a class of equipment.
- b) Equipment procedural element interface - This defines the interface to a procedural element that is available in the equipment element.
- c) Equipment element property - This defines the properties of the equipment, in a manner that mirrors the property specification of the RE Equipment Properties.
- d) Equipment element link - This defines the linkage between equipment, in a manner that mirrors the property specification of the Master Recipe RE Equipment Links.

5.3.2 Table overview and integrity constraints

Figure 18 shows the structure of the equipment information tables.

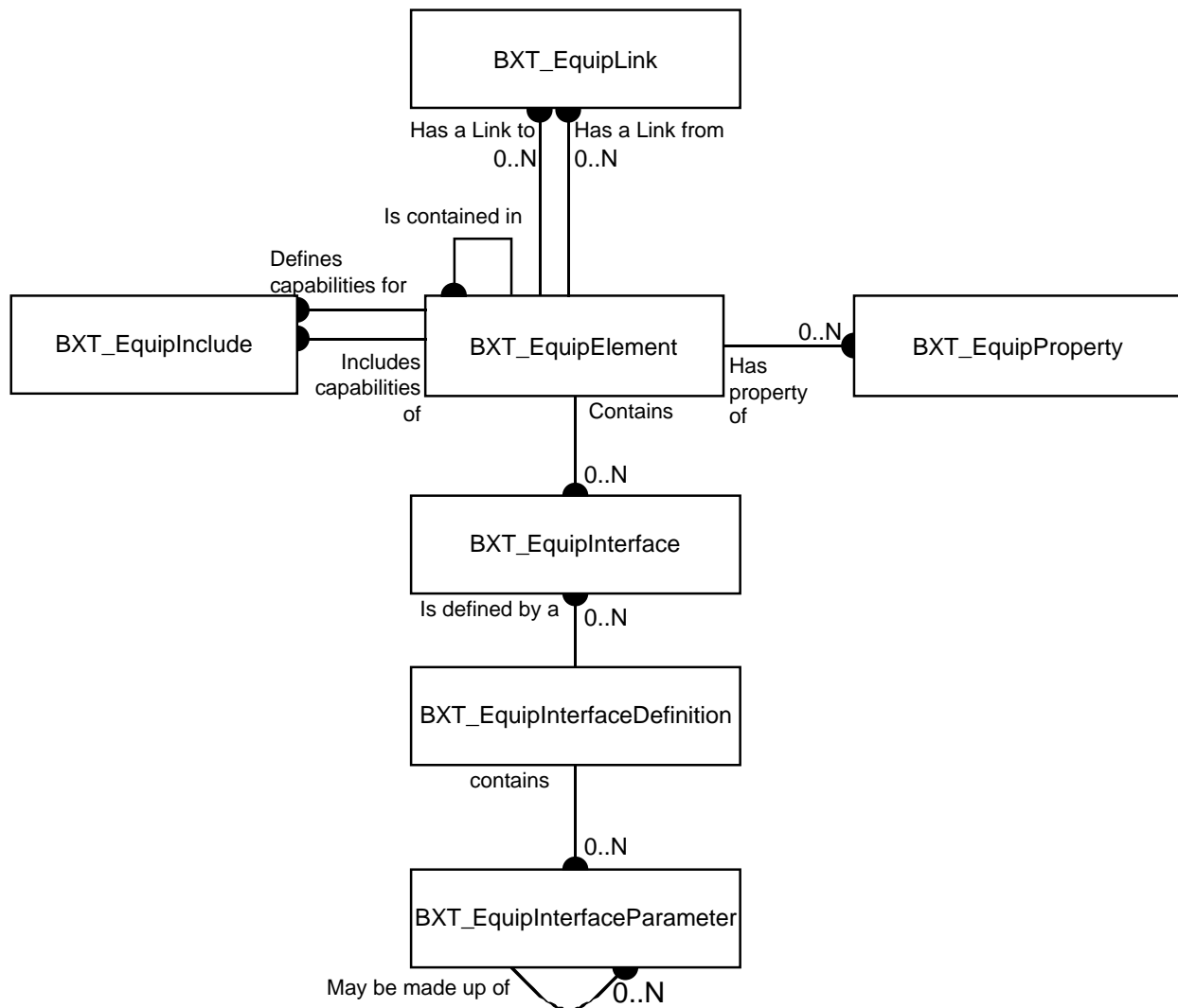


Figure 18 — Equipment information exchange tables

5.3.3 Equipment description table overview

5.3.3.1 Equipment hierarchy

The equipment information tables contain the hierarchical definition of the equipment, through the "ContainedIn" attribute of the BXT_EquipElement table. This table describes which equipment is contained within another piece of equipment. For example, a process cell equipment element may contain equipment elements that are composed of units and/or equipment modules.

Each level may have associated property specifications, data specifications, and link specifications. Each level may also have associated procedural elements in order to support the full Part 1 model, but these will typically be associated with units or equipment modules.

5.3.3.2 Equipment classes

The equipment information tables allow the specification of equipment classes through the "include the capability of" association between elements. An equipment element may also include the capability that is defined in one or more equipment elements.

For example, a unit may be a member of a unit class because it implements a set of procedures that the class defines, because it contains a set of properties of the class, or any combination of this information. One record exists in the BXT_EquipElement table for each equipment instance definition and one record exists in the BXT_EquipElement table for each class definition.

5.3.4 Equipment information table summary

The tables that are used for equipment information exchange are listed in table 46.

Table 46 — Equipment information exchange tables

TABLE NAME	FULL NAME	DESCRIPTION
BXT_EquipElement	Equipment Element	One record for each element of equipment or class of equipment.
BXT_EquipLink	Equipment Element Link Specification	One record for each link between equipment elements.
BXT_EquipInclude	Equipment Element Includes	One record for each equipment element that belongs to a class of equipment elements.
BXT_EquipProperty	Equipment Element Property Specification	One record for each property specification and its value for an equipment element.
BXT_EquipInterface	Equipment Procedural Element Interface	One record for each equipment procedural element that is defined within an equipment element.
BXT_EquipInterfaceDefinition	Equipment Procedural Element Interface Definition	One record for each equipment procedural element interface class that is defined within an equipment element.
BXT_EquipInterfaceParameter	Equipment Procedural Element Interface Parameter	One record for each data element input to, or output from, the equipment procedural element that is defined within an equipment element.

5.3.5 Equipment table definitions

The following subclause contains the detailed specifications of the equipment information tables.

5.3.5.1 Equipment element

The BXT_EquipElement table contains one record for each equipment entity (i.e., process cell, unit, equipment module, control module) (see table 47). The BXT_EquipElement table contains the definitions of entities and classes of entities (e.g., "Reactor101", "Filter20", "Reactor", "Filter"). The BXT_EquipInclude table that is defined below is used to contain the class relationships.

Table 47 — BXT_EquipElement

ATTRIBUTE	DESCRIPTION
<i>EquipmentID</i>	Identifies an equipment element or class of equipment.
EE_Type	Identifies the type of record as an enumeration (i.e., Class definition or Element definition), from the enumeration "EquipmentType."
EE_Level	Identifies the equipment level as an enumeration (i.e., Area, Unit, Equipment Module, Control Module), from the enumeration "EquipmentLevel."
ContainedIn	Identifies the equipment element that this equipment is contained in (e.g., the process cell that a unit is contained in). This field may be NULL if the equipment is not contained in any equipment or if the containing equipment is not defined.
Description	Describes the equipment element.

5.3.5.2 Equipment element links

An equipment element may have an equipment linkage specification (e.g., that equipment may feed into other equipment). These links typically define material transfers that the process cell (or units) supports. For example, a recipe may specify that a MIXER links to a REACTOR. The BXT_EquipLink table contains the equipment linkage (see table 48).

Table 48 — BXT_EquipLink

ATTRIBUTE	DESCRIPTION
<i>EquipmentID</i>	Identifies an equipment element or class of equipment.
<i>ToEquipmentID</i>	Identifies an equipment element or class of equipment that the "EquipmentID" links to.
Description	Describes the type of the equipment element link.

5.3.5.3 Equipment element includes capability

An equipment element may include the capabilities of one or more equipment element classes. For example, an equipment element may exist that represents a class of reactors with a specific set of procedural elements that represent the processing capability of reactors. A specific unit would include the capability of the reactor class and it may add additional capabilities or specifications. The BXT_EquipInclude table contains the relationship between the equipment class and the equipment instance (see table 49).

Table 49 — BXT_EquipInclude

ATTRIBUTE	DESCRIPTION
<i>EquipmentID</i>	Identifies an equipment element or class of equipment.
<i>ClassEquipmentID</i>	Identifies a class of equipment within which the EquipmentID is contained.
Description	Describes the association.

5.3.5.4 Equipment element properties

Equipment element properties define the capability that is available within an equipment element. This is done by specifying what equipment is available, in the same form as is specified in the recipe requirements, as a specification definition and the value for the specification.

The specifications may be associated with any Equipment Element in the equipment hierarchy and they define the specifications that are applicable at that level (e.g., at the process cell, unit, or equipment module level).

The BXT_EquipProperty table (see table 50) contains one record for each property that the equipment element has available (e.g., "lining type" and "glass lined unit", "size" and "50,000 gallon"). Specifications may also include equipment data elements that are available from the equipment (e.g., "SteamTemperature" from a heating equipment module, "VesselPressure" from a unit). The data elements could be available for use in recipe transitions and expressions.

Table 50 — BXT_EquipProperty

Attribute	Description
<i>EquipmentID</i>	Identifies an equipment element or class of equipment.
<i>PropertyID</i>	Identifies a property that this equipment supplies (e.g., "Lining Type," "Size," "Heat Capability," "Steam Temperature").
PropertyValue	Identifies the value for the property (e.g., "Glass," "50000," "650").
EngrUnits	Specifies engineering units of the property (e.g., "gallons," "BTU/hr").
Description	Describes the type of the equipment element property.

5.3.5.5 Equipment procedural element interface

The BXT_EquipInterface table (see table 51) contains one record for each equipment element procedure that is defined within an equipment element. Each BXT_EquipInterface record provides a mapping to the interface definition for the associated equipment element procedure. Because multiple BXT_EquipInterfaces (e.g., equipment phases) may have exactly the same external interface definition, they may all reference a single BXT_EquipInterface interface definition. This structure allows the definition of functionally equivalent BXT_EquipInterfaces, in order to allow for class-based recipes.

Table 51 — BXT_EquipInterface

ATTRIBUTE	DESCRIPTION
<i>EquipmentID</i>	Identifies an equipment element or class of equipment.
<i>EPI_ID</i>	Identifies the equipment procedural element interface of an equipment procedural element.
<i>EPI_Definition</i>	Identifies the definition of the BXT_EquipInterface.
Description	Describes the type of the equipment procedural element.

5.3.5.6 Equipment procedural element interface definition

The BXT_EquipInterfaceDefinition table (see table 52) contains one record for each equipment procedural element interface that is defined. The BXT_EquipInterfaceDefinition table contains the definition of an BXT_EquipInterface's input and output parameters.

Table 52 — BXT_EquipInterfaceDefinition

ATTRIBUTE	DESCRIPTION
<i>EPI_Definition</i>	Identifies the equipment procedural element interface definition.
Description	Describes the expected behavior of the BXT_EquipInterface.

5.3.5.7 Equipment procedural element interface parameter

The BXT_EquipInterfaceParameter table (see table 53) contains one record for each data element that is required by, generated by, or modified by an execution of the equipment procedural element that is defined within an equipment element. The BXT_EquipInterfaceParameter table contains the definition of the type and units of the required data element, and an optional reference to an enumeration set.

The BXT_EquipInterfaceParameter table also indicates if the input value is scaleable and it provides a default value that would be used if no actual value is passed to the equipment procedural element.

Table 53 — BXT_EquipInterfaceParameter

ATTRIBUTE	DESCRIPTION
<i>EPI_Definition</i>	Identifies the equipment procedural element interface class.
<i>ParameterID</i>	Identifies the name of the parameter that is used by the equipment procedural element.
ParentParamID	Identifies the parent parameter set of which this parameter is a member, and this field is NULL if there is no parameter set.
Type	Identifies the type of the data that is used by the equipment procedural element, from the enumeration set ValueDataType.
EngrUnits	Identifies the engineering units of the data that is used by the equipment procedural element.
EnumSet	Identifies the enumeration set of which this element is a member (i.e., if not NULL).
Scaled	Specifies if the parameter may be scaled before it is passed to the equipment procedural element. If this field is TRUE, the parameter may be scaled, from the enumeration set "Boolean."
DefaultValue	Identifies the default value that is used if no value is passed to the equipment procedural element.
Description	Describes the equipment procedural element parameter.

5.4 Schedule information exchange

The schedule tables define one or more scheduled batches for a given process cell. Each scheduled batch contains additional batch specific information that may be used along with the master recipe information, in order to create a control recipe. Each scheduled batch may define a set of parameter values that are needed for recipe creation and a set of equipment requirements. Not all information that is needed to execute a control recipe needs to be provided in the schedule information exchange tables; additional information may be supplied by the control system or operator.

Scheduling activities may require information in addition to that provided by these tables. Scheduling may need to use other means in order to obtain some of the data (e.g., the status of equipment, material inventories, utility conditions). This information is best exchanged by other methods.

5.4.1 Schedule table overview

The schedule information exchange tables allow for information on multiple scheduled batches to be included in a single set of tables.

The schedule information exchange tables do not specify how the information is created or how it is used. Tools that use the information may include scheduling packages, batch automation packages, operational display packages, and work-order tracking packages. The importing and exporting tools shall determine the correct use of schedule information in the tables.

Figure 19 shows the five tables that make up the schedule information exchange tables.

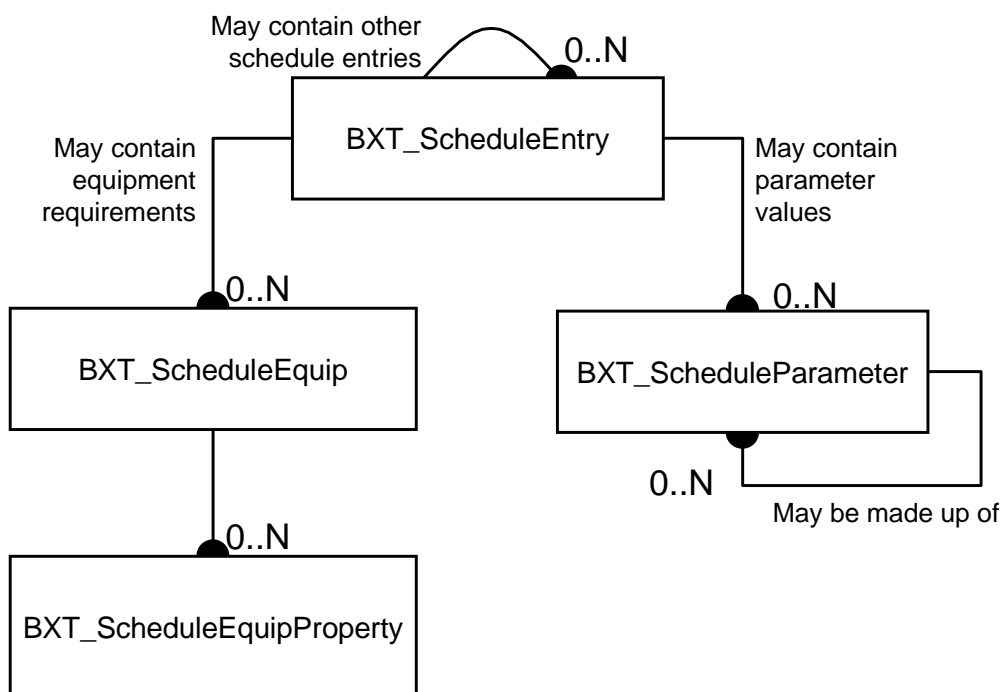


Figure 19 — Schedule structure

5.4.2 Schedule table summary

The tables that are used for schedule information exchange are listed in table 54.

Table 54 — Schedule information exchange tables

TABLE NAME	DESCRIPTION
BXT_ScheduleEntry	One record for each scheduled item (i.e., a batch, unit recipe, or cleaning event).
BXT_ScheduleEquip	One record for each equipment selection requirement. Permissible equipment selections may be defined.
BXT_ScheduleProperty	One record for each property specification for each equipment requirement.
BXT_ScheduleParameter	One record for each parameter item of the schedule record.

5.4.3 Schedule table definitions

5.4.3.1 Schedule entry

The BXT_ScheduleEntry table (see table 55) contains one element for each scheduled event. Schedule entries can represent a batch or some other processing activity (e.g., a unit recipe). If the scheduled event is a batch, the table contains the batch identification that is associated with the scheduled batch (i.e., if assigned at this time) in the ScheduleEntryID field and the master recipe that is associated with the scheduled batch is identified in the RE_ID field. The ScheduleEntryString field is used to uniquely identify scheduled entries when their "real" ID has not yet been assigned.

Table 55 — BXT_ScheduleEnTRY

ATTRIBUTE	DESCRIPTION
<i>ScheduleEntryID</i>	Identifies the unique ID (within this table) of the schedule entry. This may be a campaign ID, batch ID, unit procedure ID, or unique string with no external meaning.
ParentSchedID	Identifies the parent scheduled item record to which this record is related by using the parent's ScheduleEntryString.
ExternalID	Defines the identifier that is used by the business to identify this schedule entry.
<i>RE_ID</i>	Identifies the Recipe Element (e.g., <i>Red Oak</i>). When combined with the "version," this field defines a unique instance of a RE. When the record represents the master recipe, this field contains the master recipe ID. (Identifies the Recipe Element that is referenced by this schedule entry.)
<i>REVersion</i>	Identifies the version of the master recipe.
SE_Type	Defines the type of entity that this schedule record represents, from the enumeration set "SE_Type". This definition allows a batch record to have more detailed schedule information at lower levels in the procedure hierarchy (i.e., property and equipment requirements can be identified for each unit procedure in the recipe), as well as handle the scheduling of campaigns and groups of batches.
BatchID	Identifies the ID of the batch for which this schedule item is a part.
LotID	Identifies the ID of the lot for which this schedule item is a part.
CampaignID	Identifies the ID of the production campaign for which this schedule item is a part.
ProductID	Identifies the product to be made.
OrderID	Identifies the production order or customer order(s) with which this schedule record is related.
SE_Action	Defines the expected action by the receiving tool, as an enumeration (i.e., New, Update, Delete, User defined), from the enumeration set "ScheduleAction."
SchedStatus	Defines the status of the schedule record (i.e., Complete, In-process, Scheduled, Schedule Hold), from the enumeration set "ScheduleStatus."
StartCondition	Specifies the expected starting condition of the schedule record, if known (e.g., "Starts before...", "Follow...").
InitialMode	Defines the mode in which the schedule record begins execution as an enumeration (i.e., Automatic, Semi-automatic, Manual), from the enumeration set "ScheduleMode."
SchedStartTime	Specifies the expected starting time of the schedule record, if known.
SchedEndTime	Specifies the expected end time of the schedule record, if known.
BatchPriority	Specifies a priority that is placed on the schedule record, if known. Lower numbers have higher priority (e.g., Priority 1 is more important than Priority 7).
BatchSize	Defines the requested size, or scale factor, for the batch, based on the scale factor for the batch, as defined in the master recipe.
EngrUnits	Optionally identifies the engineering units of measure for the BatchSize.
SENote	Provides information or instructions to operations.
Description	Describes the scheduled item and/or product (e.g., Premium Beer).

5.4.3.2 Schedule record equipment requirement

The BXT_ScheduleEquip table (see table 56) contains one element for each equipment requirement for a schedule record. The related BXT_ScheduleEquipProperty table contains a specific property definition that this equipment needs to satisfy.

Typical requirements for a batch may be equipment selections. A schedule record's equipment requirements and properties will generally correspond to a master recipe's equipment requirements and properties. For example, a scheduling package may specify a specific unit to be used in production. The BXT_ScheduleEquip would specify the equipment identity as defined in a RE and the BXT_ScheduleProperty would specify the name of the selected unit.

Table 56 — BXT_ScheduleEquip

ATTRIBUTE	DESCRIPTION
<i>ScheduleEntryID</i>	Identifies the scheduled item that is exchanged.
<i>RequirementID</i>	Provides a unique name for the equipment requirement of a scheduled item. This name may refer to an individual piece of equipment, a class of equipment, a list of allowable equipment, or other groups of equipment (e.g., a train).
Description	Describes the requirement (e.g., First reactor unit in the batch).

5.4.3.3 Schedule record equipment property requirement

The BXT_ScheduleProperty table (see table 57) contains one element for each property specification for each equipment requirement for a schedule record.

Because a single equipment requirement for a given scheduled item can have multiple property criteria (e.g., material of construction and volume), a separate table is used to list these requirements.

Table 57 — BXT_ScheduleProperty

ATTRIBUTE	DESCRIPTION
<i>ScheduleEntryID</i>	Identifies the scheduled item exchanged.
<i>RequirementID</i>	Identifies the requirement set associated with the batch, typically an equipment or material class.
<i>PropertyName</i>	Identifies the name of the property for the schedule batch.
PropertyValue	Specifies the value for the property for the schedule batch.
EngrUnits	Optionally identifies the engineering units of measure for the PropertyValue.
Description	A description of the property (e.g., Use UNIT345 as the first reactor unit in the batch).

5.4.3.4 Schedule parameter

The BXT_ScheduleParameter table (see table 58) contains one element for each parameter for a scheduled item. Parameters in a schedule item are typically parameters in the master recipe, but they may also be information for the operator or for other users of the schedule information.

Limits for schedule parameter items may be defined in the same way that sub-parameters further define the parameters in recipes.

Table 58 — BXT_ScheduleParameter

ATTRIBUTE	DESCRIPTION
<i>ScheduleEntryID</i>	Identifies the scheduled item that is exchanged.
<i>ParameterID</i>	Identifies the parameter for the scheduled batch.
ParentParameterID	Identifies the parent parameter set of which this parameter item is a member (i.e., NULL if there is none).
ParameterValue	Specifies the value for the parameter for the scheduled batch.
EngrUnits	Optionally identifies the engineering units of measure for the ParameterValue.
ItemLocation	Defines where in the recipe structure this parameter item is applied. This record is used when the ParameterID is a simple alias, and the ParameterID does not provide sufficient information to identify where it is applied.
EnumSet	Identifies the enumeration this element is a member of (i.e., if not NULL).
Description	Describes the parameter for the related scheduled item and/or product. (e.g., Premium Beer).

5.5 Production information exchange

Production information exchange provides a structure in order to exchange all of the information about the execution of a batch.

Many tools (e.g., batch automation systems, laboratory information management systems, batch reporting systems, batch analysis systems, scheduling systems, simulation systems) could use this information.

The structure of the exchange tables allows for data on multiple batches to be exchanged in the same tables.

Production information is contained in three areas:

- a) the control recipe;
- b) the equipment that the recipe ran against; and
- c) a log of all events that occurred during the execution of the recipe.

5.5.1 Control recipe information

The control recipe information may be exchanged using the MR tables, with the ProductID as the representation of the batch identification.

Because the control recipe starts as a copy of a master recipe, the MR tables may contain the master recipe used for production and the event table may contain any changes to the control recipe, or the MR tables may contain the control recipe after modification. In either case, the RecipeID and REVersion information serve to identify the specific master recipe that was used to create the control recipe.

5.5.2 Equipment information

The equipment that is used in the production of the batch may be exchanged using the BXT_EquipElement (Equipment Entity) tables. The tables may contain the definition of the entire process cell that was used in production, or they may only contain the subset of equipment actually that was used in the production of the batch.

5.5.3 Batch history

The batch history is contained in two tables, the BXT_HistoryElement table and the BXT_HistoryLog table.

The BXT_HistoryElement table is the record of each execution of a recipe procedural element and/or the equivalent equipment procedural element. This table contains one element for each RE or EPE execution. Figure 20 shows the elements of batch history and the relationship to the previously defined equipment element tables. The BXT_HistoryElement table (see table 59) contains records that may be used to reference the associated equipment and equipment procedural element.

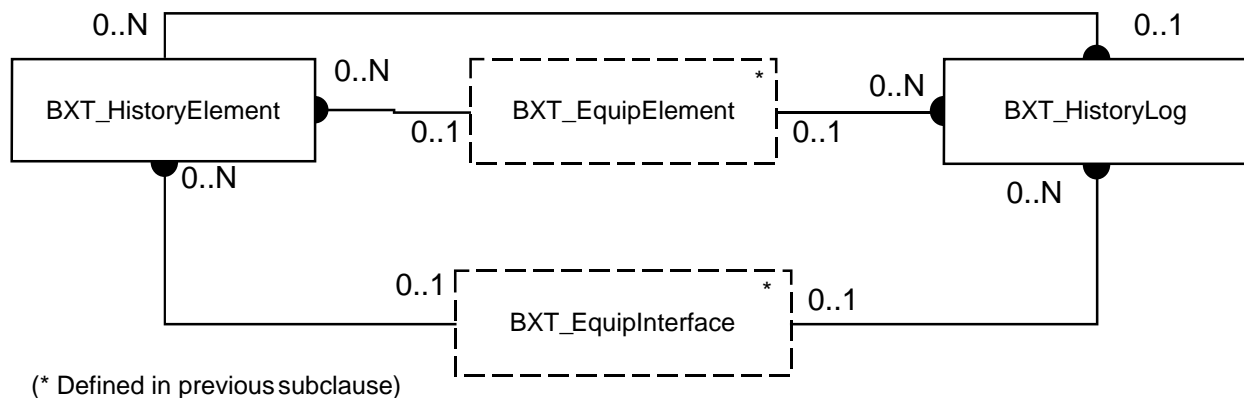


Figure 20 — Batch history

The BXT_HistoryLog and BXT_HistoryElement tables contain a list of time-defined records that occurred during the production of the batch, one record for each recorded event in the batch. The BXT_HistoryElement tables contains each instance of use of a recipe procedural element or equipment procedural element. The BXT_HistoryLog table contains one record for each event that occurred for the procedural element (e.g., the start of execution of the procedural element, a mode change, a state change, a value that is reported from the element).

The batch history tables are designed with several objectives:

- Information that would be duplicated in multiple records is moved to the BXT_HistoryElement table and this provides a significant amount of reduction in the size of the BXT_HistoryLog table.
- Information that can be described in the equipment tables is only referenced by key value. The BXT_HistoryElement record contains an equipment element identification to the associated equipment and an identification of the associated equipment procedural element.
- The BXT_HistoryLog table also includes a reference to the equipment element and equipment procedural element, in order to simplify the use of the BXT_HistoryLog table, even though this information is duplicated in the BXT_HistoryElement table.

5.5.3.1 History element

The BXT_HistoryElement table contains one record for each recipe element (see table 59).

Table 59 — BXT_HistoryElement

ATTRIBUTE	DESCRIPTION
<i>HistoryElementID</i>	Provides a generated identification number that is required for relational integrity.
BatchID	Specifies a unique identification of the batch that is associated with the BXT_HistoryElement record. This is duplicated information with the BXT_HistoryLog table record.
MasterRecipeID	Specifies the identification of the master recipe that is associated with the batch.
MasterRecipeVersion	Specifies version identification of the related master recipe.
ControlRecipeID	Identifies the control recipe ID. In some cases, this identification may be different from the batch ID.
ReferenceEquipProcedure	Identifies whether the procedural control hierarchy is defined to be referring to recipe or equipment, as a Boolean enumeration, with True identifying a reference to equipment.
RecipeProcedure	Identifies the Procedure that is associated with the BXT_HistoryElement record.
UnitProcedure	Identifies the Unit Procedure that is associated with the BXT_HistoryElement record.
UnitProcedureCounter	Specifies an instance of execution counter, which is the number of times that the unit procedure has been executed. This counter is needed because the recipe may execute the same unit procedure multiple times due to manual intervention or looping.
Operation	Identifies the Operation that is associated with the BXT_HistoryElement record.
OperationCounter	Specifies an instance of execution counter, which is the number of times that the unit procedure has been executed. This counter is needed because the recipe may execute the same operation multiple times due to manual intervention or looping.
Phase	Identifies the Recipe Phase that is associated with the BXT_HistoryElement record.
PhaseCounter	Specifies an instance of execution counter, which is the number of times that the unit procedure has been executed. This counter is needed because the recipe may execute the same phase multiple times due to manual intervention or looping.
EquipmentID	Identifies the equipment element that may be associated with the BXT_HistoryElement record. This is duplicated information with the BXT_HistoryLog table record.
EPI_ID	Identifies the equipment procedural element that may be associated with the record. This is duplicated information with the BXT_HistoryLog table record.

5.5.3.2 History Log

The BXT_HistoryLog table (see table 60) contains five sets of information about the event that should be logged:

- a) the time of the event;

- b) the batch and recipe information that is associated with the event;
- c) the equipment that is associated with the event;
- d) the operator that is associated with the event; and
- e) the event information.

The BXT_HistoryLog contains production information events. The production information events are categorized using enumerations in the RecordSet and RecordSubSet fields to aid information handling (e.g., filtering and sorting).

Table 60 — BXT_HistoryLog

ATTRIBUTE	DESCRIPTION
<i>RecordID</i>	Specifies a generated identification number that is required for relational integrity.
UTC	Identifies the Universal Coordinated Time (UTC) and date of the record.
LocalTime	Identifies the local time and date of the record.
BatchID	Provides a unique identification of the batch that is associated with the record.
HistoryElementID	Provides a unique identification of the instance of execution or the associated recipe element or equipment procedural element that may be associated with the record. This field is a key into the BXT_HistoryElement table.
EquipmentID	Identifies an equipment element that may be associated with the record.
EPI_ID	Identifies the equipment procedural element that may be associated with the record.
UserID	Specifies the name of the user, if any, who is associated with the record.
RecordSet	Specifies the type of the record, from the RecordSet enumerations.
RecordSubSet	Specifies the subtype of the record, from the enumeration set that is specified by the RecordSet record.
RecordAlias	Defines an equipment-independent record specification (e.g., "vessel top temperature").
NewValue	Specifies the data value that is associated with the record type and subtype.
OldValue	Defines a field that may contain the previous data value.
EngrUnits	Specifies the engineering units, if any, that are appropriate for the NewValue and OldValue.

5.6 Exchange table domains

The same information is contained in multiple tables for key fields and many of the fields in the tables have the same domain (i.e., specific data type and range). The ISO/IEC standard for SQL does not define domains, so table 61 contains the definition of the domains for selected table attributes in the exchange tables.

Table 61 — Exchange table domains

DOMAIN NAME	TYPE	DESCRIPTION
BXT_MRecipeElement-Author	CHAR (32)	Name or ID of the author.
BXT_MRecipeStep-StepID BXT_MRecipeStepParameter-StepID BXT_MRecipeStepEquip-StepID BXT_ScheduleEntry-StepID	CHAR (128)	Identifies a step within a recipe procedural element.
BXT_MRecipeElementParameter-EngrUnits BXT_MRecipeStep-ScaleEngrUnits BXT_EquipProperty-EngrUnits BXT_EquipInterfaceParameter-EngrUnits BXT_ScheduleEntry-EngrUnits BXT_ScheduleProperty-EngrUnits BXT_ScheduleParameter-EngrUnits BXT_HistoryLog-EngrUnits	CHAR (32)	Engineering Units Specification.
BXT_MRecipeElement-Status BXT_ScheduleEntry-SchedStatus	INTEGER	Status of the recipe or procedural element as an enumeration.
BXT_MRecipeElement-RE_ID BXT_MRecipeElementParameter-RE_ID BXT_MRecipeStep-ParentRE BXT_MRecipeStep-RE_ID BXT_MRecipeStepParameter-ParentRE BXT_MRecipeTransition-RE_ID BXT_MRecipeLink-RE_ID BXT_MRecipeElementEquip-RE_ID BXT_MRecipeOtherInformation-RE_ID BXT_MRecipeStepEquip-ParentRE	CHAR (128)	Identifies a recipe procedural element.
BXT_MRecipeElement-REVersion BXT_MRecipeElementParameter-REVersion BXT_MRecipeStep-ParentVersion BXT_MRecipeStep-REVersion BXT_MRecipeStepParameter-ParentVersion BXT_MRecipeTransition-REVersion BXT_MRecipeLink-REVersion BXT_MRecipeElementEquip-REVersion BXT_MRecipeStepEquip-ParentVersion BXT_MRecipeOtherInformation-REVersion BXT_ScheduleEntry-Version BXT_HistoryElement-MasterRecipeVersion	CHAR (16)	Version identifier for all elements with versions.
BXT_MRecipeElementParameter-ParameterID BXT_MRecipeElementParameter-ParentParamID BXT_MRecipeStepParameter-ParameterID BXT_MRecipeStepParameter-ParentParamID BXT_EquipInterfaceParameter-ParameterID BXT_EquipInterface-ParentParamID BXT_ScheduleParameter-ParameterID	CHAR (32)	Identifies a parameter of an RE, definition of use.

Table 61 — Exchange table domains (continued)

DOMAIN NAME	TYPE	COMMENTS
BXT_MRecipeElement-ProcessCellID	CHAR(32)	Identifies a process cell or other equipment entity.
BXT_EquipElement-EquipmentID		
BXT_EquipLink-EquipmentID		
BXT_EquipLink-ToEquipmentID		
BXT_EquipInclude-EquipmentID		
BXT_EquipInclude-ClassEquipmentID		
BXT_EquipProperty-EquipmentID		
BXT_EquipInterface-EquipmentID		
BXT_EquipInterfaceParameter-EPI_Definition		
BXT_ScheduleProperty-RequirementID		
BXT_ScheduleEquip-RequirementID		
BXT_HistoryLog-EquipmentID		
BXT_HistoryElement-EquipmentID		
BXT_ScheduleEquip-RequirementID		
BXT_HistoryLog-EquipmentID		
BXT_HistoryElement-EquipmentID		

6 Procedure function charts

ISA hereby grants a non-exclusive, royalty-free, limited license under ISA's copyright in the standard, to copy, display and distribute this section of this ISA standard (including software included in or defined by such section), as follows:

1. Producers of products or services intended to comply with the standard may incorporate this designated section, but only to the extent reasonably necessary to make, use, and distribute any product or service (including product documentation) that is compliant with this standard.
2. End users of a product or service made by a producer acting under the preceding license may reproduce and use the designated section, but only to the extent reasonably necessary to enjoy the intended functions of the product or service and to maintain, configure, or reconfigure systems to be compliant.
3. Persons providing education on or promotion of the standard may copy, display and distribute the designated section, but only to the extent reasonably necessary to provide information related to the standard.

Except as expressly permitted, all other reproduction and distribution without permission of ISA is prohibited. All copies of this section of the standard made or distributed under this license must cite the standard and include the following notice of copyright:

Copyright © 2001 by ISA—The Instrumentation, Systems, and Automation Society. All rights reserved. Used with permission of ISA.

This clause defines a method for graphical representation of Master and Control recipes. The representation of the procedure is called a Procedure Function Chart (PFC). This clause also addresses requirements for representation of formula, equipment requirements, header, and other information.

The PFC language, as defined in this standard, is designed to support recipes with complex procedures (e.g., parallel steps, selections) that vary from one product to another.

Procedure Function Charts build upon Function Charts, as defined in IEC 60848. In Procedure Function Charts, it is presumed that steps follow transitions and transitions follow steps, as described in IEC 60848. However, there are significant differences between that standard and this one that are necessary in order to meet the requirements of procedural control, as opposed to documentation. In batch control, equipment procedural elements contain the procedural logic that regulates when and how they complete. In addition, many equipment phases in batch control are designed to complete a task and then terminate. The PFC approach supports the separation of recipe procedural elements from equipment procedural elements, by recognizing that equipment procedural elements, once started, execute independently. Another difference that must be addressed in a procedure function chart is the multiple level structure of recipe procedural elements; therefore, it must be clear whether a symbol represents a recipe phase, a recipe operation, a recipe unit procedure, or an entire recipe procedure.

6.1 Procedure function chart notation

Procedure function charts depict procedural logic, using a series of symbols that are interconnected by directed links, in order to define the execution sequence of procedural elements. The execution of procedural elements may occur in series or parallel and the execution may be dependent upon conditional logic. Activities that are depicted include the intended execution of unit procedures, operations, phases, and the evaluation of transitions. In general, the flow of execution is top to bottom and left to right. Procedure function charts are used to depict the procedural logic for all levels of the recipe: recipe procedure, recipe unit procedure, and recipe operation.

6.1.1 Symbols

A procedure function chart is defined by a set of symbols for

- a) elements (i.e., recipe procedural elements);
- b) begin and end points;
- c) resource allocation;
- d) element synchronization;
- e) recipe transitions; and
- f) basic structures (i.e., directed links, sequence selection, simultaneous sequences).

Only the global representation of the symbols is imposed; dimensions and details (e.g., thickness of lines and font of characters) are left to each implementation.

6.1.1.1 Elements

Symbols shall be used to represent a recipe phase, a recipe operation, a recipe unit procedure, or a recipe procedure. A graphical indication within the symbol shall be used to identify the symbol as representing a recipe phase, a recipe operation, a recipe unit procedure, or a recipe procedure. One example for identifying procedural elements is shown in figure 21.

Part 1 defines only four levels in the procedural hierarchy and only those four levels are described in this document. However, additional levels are possible and they may be defined for various purposes. While this document only addresses the four levels of procedural elements that are defined in Part 1, any additional levels that are defined separately may be identified either graphically, following the same general principles described herein, or textually by a string that starts in the upper left-hand corner of the rectangle.

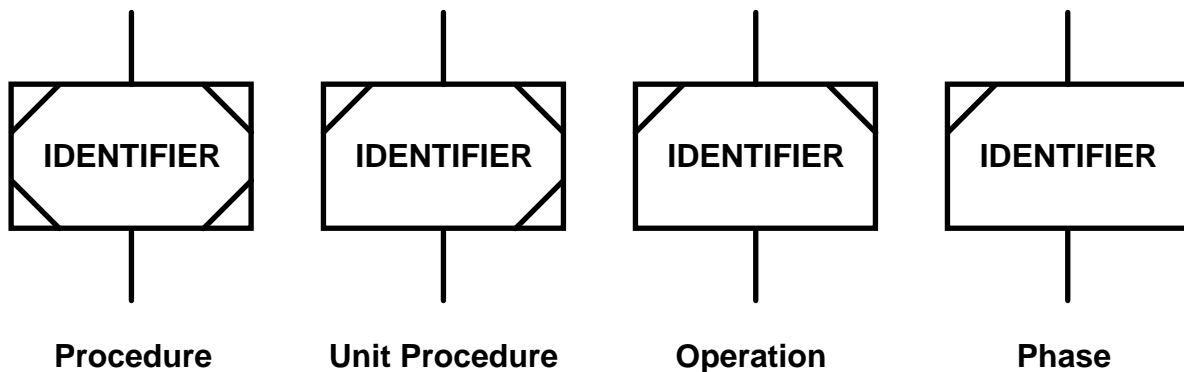


Figure 21 — Recipe procedural element symbols

A procedural element above the level of a phase may represent an encapsulation of other procedural elements at the next lower level in the procedural control hierarchy. A procedural element that represents an encapsulation, where the lower level recipe procedural elements are not shown, shall be identified by a plus sign (+) in the upper right hand corner of the rectangle, representing the procedural element (see figure 22). Procedural element symbols that expose the encapsulation shall be identified by a minus sign (-) (see figure 38). Procedural element symbols that reference an equipment procedural element shall have no indication.

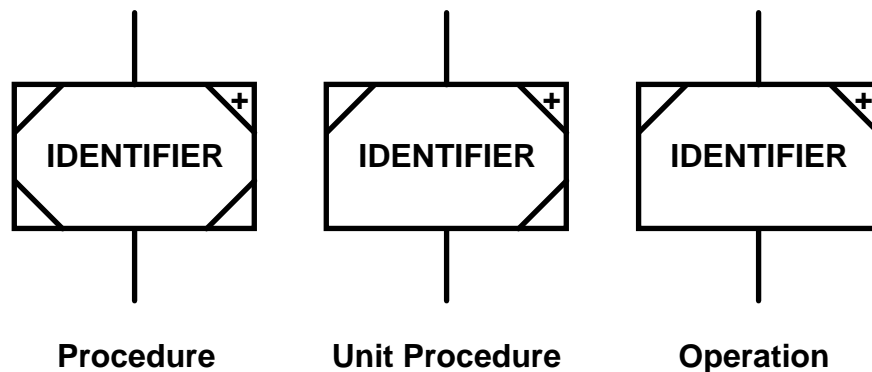


Figure 22 — Procedural elements that encapsulate lower-level recipe procedural elements

When a procedural element represents an encapsulation of subordinate procedural elements, a separate, lower-level procedure function chart that specifies the subordinate procedural elements and associated ordering symbols may be used to define it. The icon representing the encapsulating recipe procedural element may also be expanded to allow the lower level procedure function chart to be depicted within the boundaries of the encapsulating icon. The process of expansion of single symbols may continue until there is no subordinate level. An equipment procedure may possibly be shown as an expansion of the recipe procedural element that refers to it.

6.1.1.2 Begin and end points

Procedure function charts shall have at least one begin point and at least one end point, in contrast to a sequential function chart that may continuously cycle.

6.1.1.2.1 Begin

At least one begin symbol (see figure 23) shall be used to designate the beginning of each procedure function chart and/or each subordinate procedure function chart.

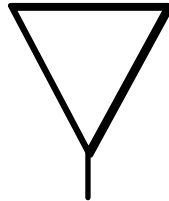


Figure 23 — Begin symbol

6.1.1.2.2 End

At least one end symbol (see figure 24) shall be used to designate the end of each procedure function chart and/or each subordinate procedure function chart.

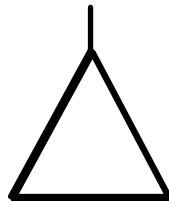


Figure 24 — End symbol

6.1.1.3 Resource allocation

Some resources are allocated to a batch before the batch can make use of them. Controlling the timing of this allocation may be important to the recipe or for scheduling; therefore, resource allocation rules and start conditions are needed. Allocation, if shown, shall be depicted by an oval icon (see figure 25) that represents the encapsulation of the resource allocation requirements for a recipe entity. The allocation symbol is a recipe procedural element appropriate to the level of the procedure function chart and it may be used at any level of the procedural hierarchy.

The allocation symbol represents the data and/or logic that determines what will be allocated to the batch (e.g., which specific unit or criteria for unit selection, equipment modules, materials, personnel) and when it will be allocated to the batch (e.g., two hours after the start of another unit procedure). The allocation symbol may contain logic that is to be executed by process management or through equipment procedural elements that are not necessarily associated with actual physical equipment. An explicit transition that follows the allocation symbol may be used to specify the starting condition for the following recipe entity.

The allocation symbol may also be used to specify explicit de-allocation. In this case, an appropriate text annotation should be used to indicate its use for de-allocation.

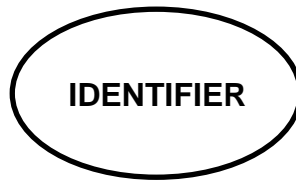


Figure 25 — Allocation symbol

6.1.1.4 Element synchronization

Synchronization between recipe elements may need to be depicted (see figure 26). Synchronization, if shown, shall be depicted by a rectangle that extends out of either side of the symbol for any of the recipe elements that are involved in the synchronization. Corresponding synchronization symbols may be connected with a dashed or other line that is distinguishable from a directed link, when the location of the symbols allows such notation without confusion. If the synchronization represents a material transfer, an arrowhead shall be added to indicate the direction of material movement intended. If no line is used, a unique identifier that identifies the specific interaction shall be provided at each recipe element that is involved in the synchronization.

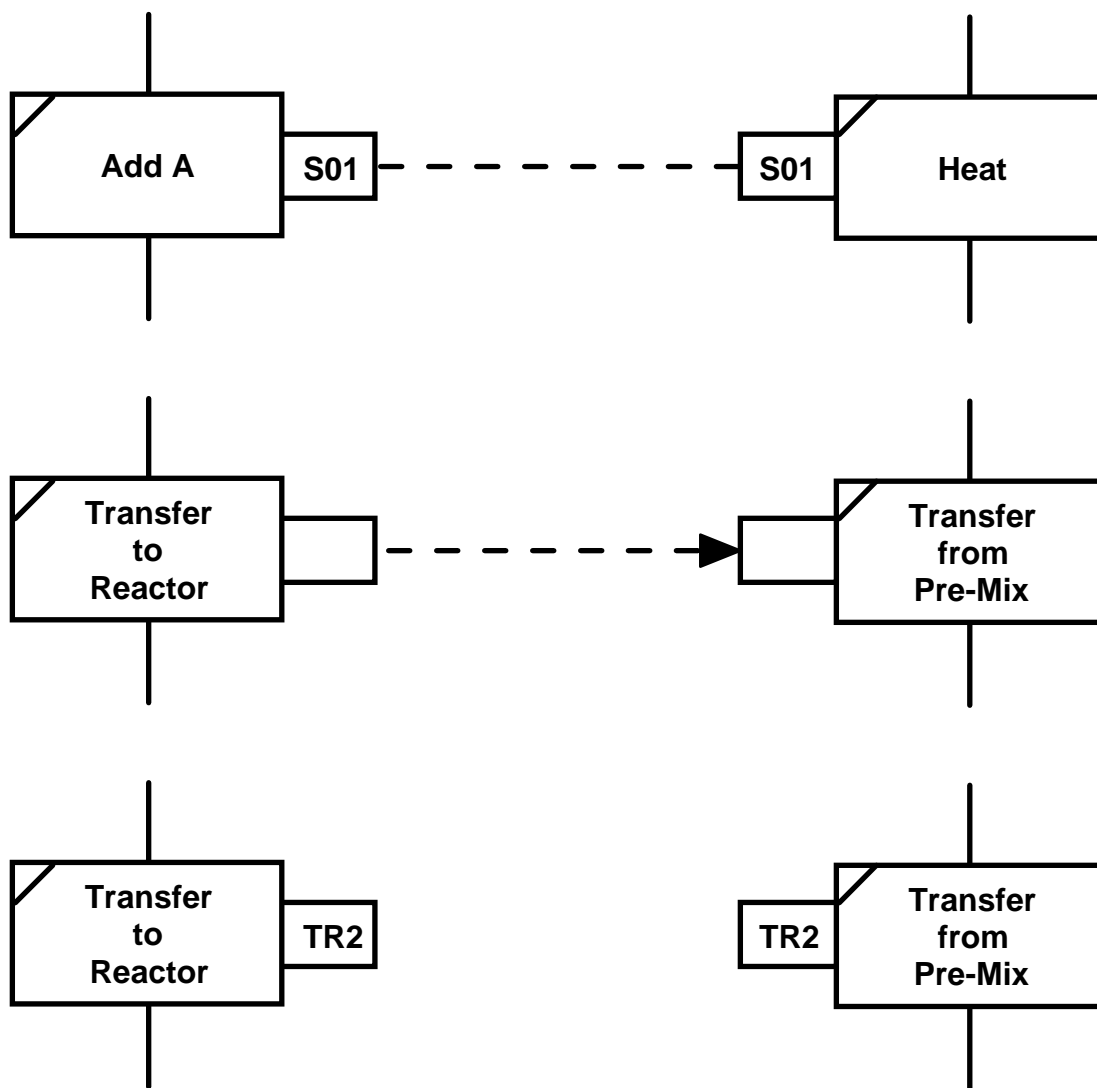


Figure 26 — Element synchronization examples

6.1.1.5 Recipe transitions

Procedure functions charts depict two types of transitions: an implicit and an explicit transition.

6.1.1.5.1 Implicit transitions

A directed link that consists of a single line between recipe entities (see figure 27) shall indicate a transition whose only condition shall be that the entities directly preceding the transition have finished their execution. No logic conditions shall be entered for this type of transition.

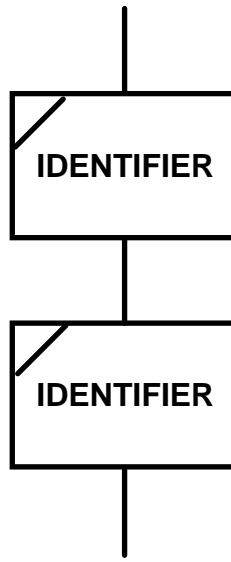


Figure 27 — Implicit transition

6.1.1.5.2 Explicit transition

A directed link that consists of a single line between recipe entities, with two short, tightly spaced double bars perpendicular to the link line (see figure 28), shall be used to indicate the explicit recipe transition.

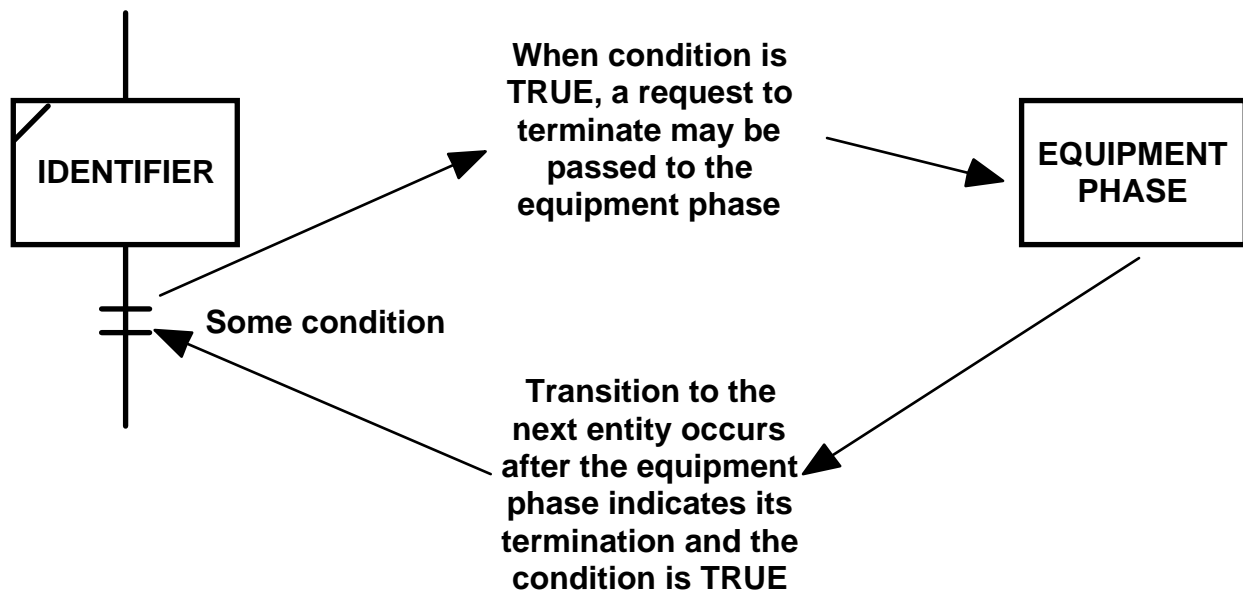


Figure 28 — Explicit transition

This transition is defined by an expression that evaluates to either true or false. The transition is continually evaluated, once the immediately preceding entity becomes active.

The transition is used to perform two functions:

- a) to interrupt the execution of a branch of the recipe procedural logic; and
- b) to request the termination of all immediately preceding procedural elements (e.g., unit procedures, operations, phases).

The termination of the immediately preceding procedural element may be a condition of the expression. However, a conditional language for the expression is not defined in this standard.

When a transition becomes true, the active procedural elements immediately preceding the transition shall be requested to terminate. When immediately preceding procedural elements terminate before the transition evaluates true, the transition shall continue to evaluate its logic until it is true.

An entity that immediately follows a transition shall be activated only after the transition condition evaluates true and the procedural elements preceding the transition terminate.

6.1.1.6 Basic structures

Structures define the intended thread of execution of the recipe elements. The simplest case is a series of recipe procedural elements that will be activated one after another. More complex structures include sequence selection and simultaneous sequences.

6.1.1.6.1 Beginning of sequence selection

The beginning of a sequence selection is shown in figure 29. Each branch of a sequence selection shall start with a transition. A selection of one path, out of several possibilities, is represented by as many transitions (i.e., under the horizontal line) as there are possibilities. Only one sequence shall be selected from the set of sequences below the line. The transitions shall be evaluated in left to right priority. The sequence below the transition that becomes true first, when evaluated in this manner, shall become the selected sequence.

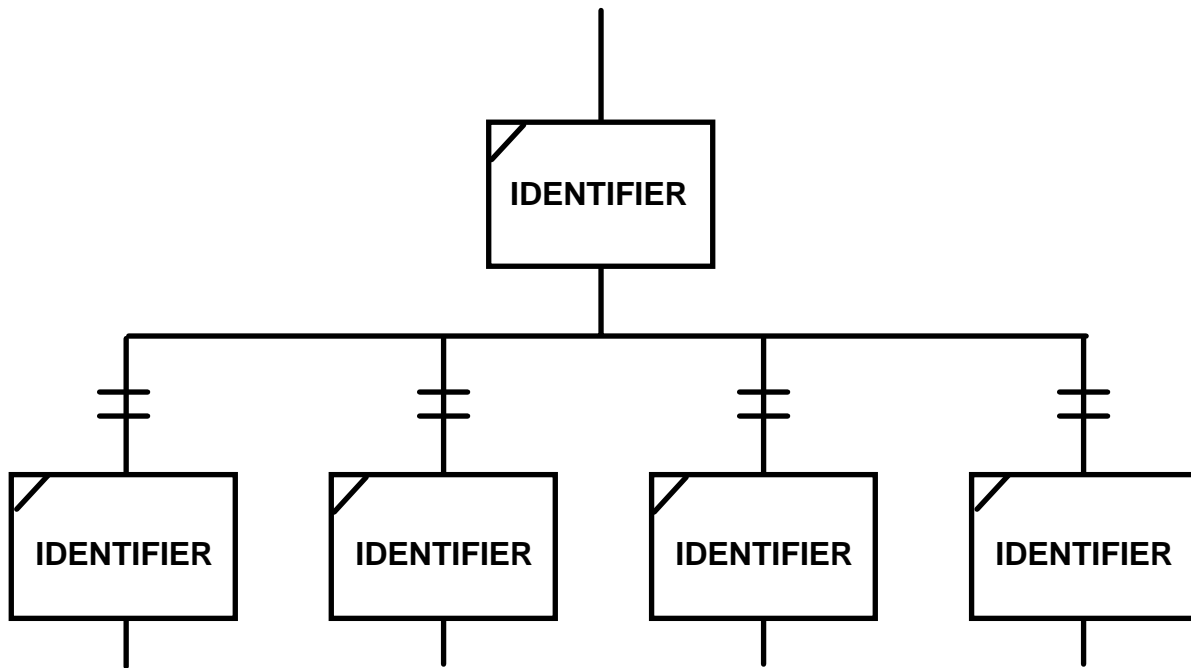


Figure 29 — Beginning of sequence selection

6.1.1.6.2 End of sequence selection

The end of sequence selection shows the joining of possible threads of execution from a sequence selection (see figure 30).

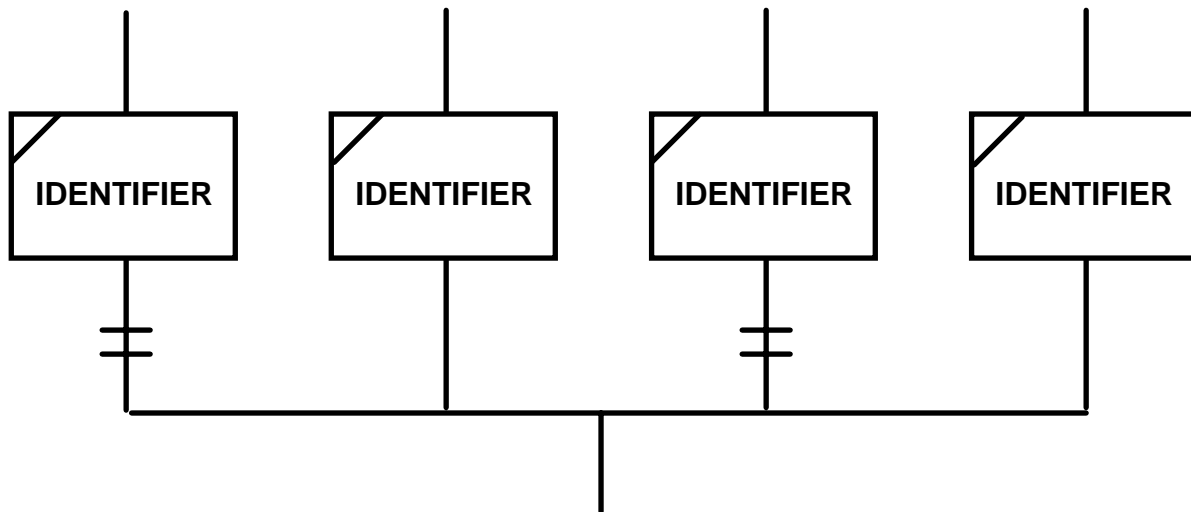


Figure 30 — End of sequence selection

6.1.1.6.3 Beginning of simultaneous sequences

The beginning of simultaneous sequences (see figure 31) shows the start of independent threads of execution of the recipe elements and there is one thread of execution for each path under a start of selection. All threads of execution shall be joined back to a single thread of execution in the recipe entity. The beginning and ending of threads of execution do not have to be matched. If an explicit transition is needed, then it shall be above the parallel lines.

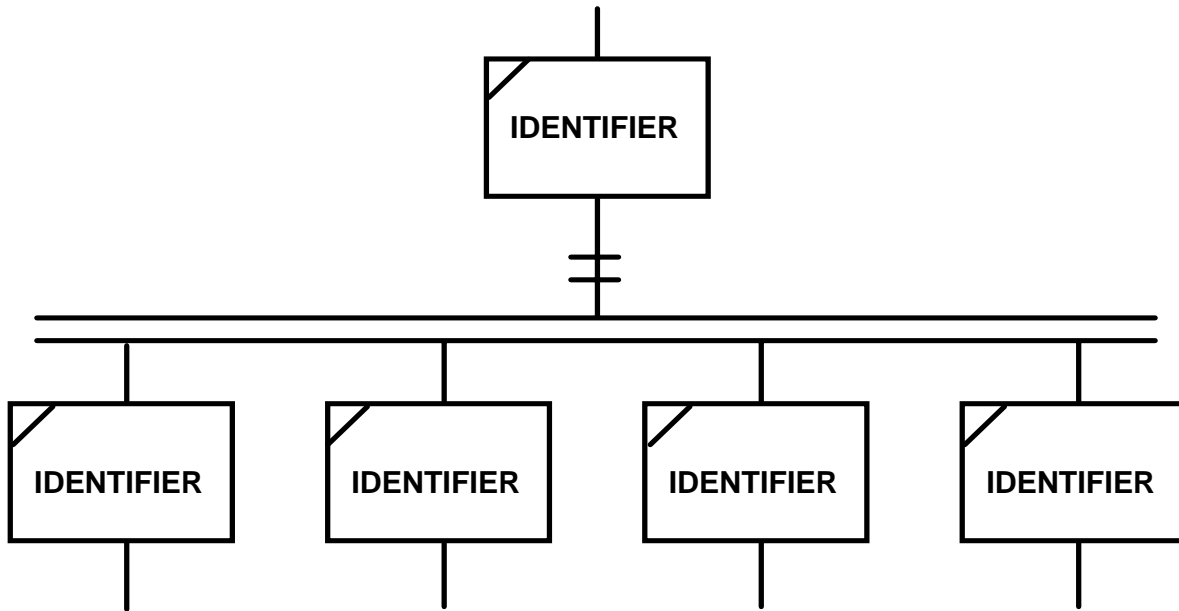


Figure 31 — Beginning of simultaneous sequences

6.1.1.6.4 End of simultaneous sequences

The end of simultaneous sequences shows the joining of independent threads of execution of the recipe element (see figure 32). The transition that immediately follows the parallel lines is evaluated only when all of the entities that immediately precede the parallel lines are either active or have completed.

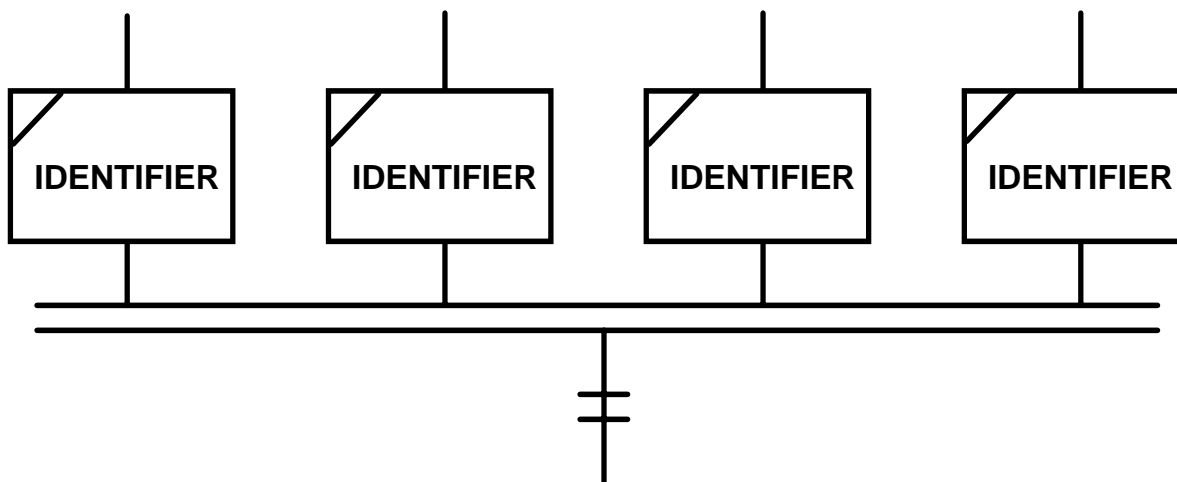


Figure 32 — End of simultaneous sequences

6.1.1.6.5 Rules for valid diagrams

Valid diagrams shall follow consistent rules for threads of execution. Independent simultaneous threads of execution shall be joined. The end of sequence selection cannot be used to join simultaneous threads of execution. Figure 33 shows an example of a valid diagram segment with sequence selection and end of sequence.

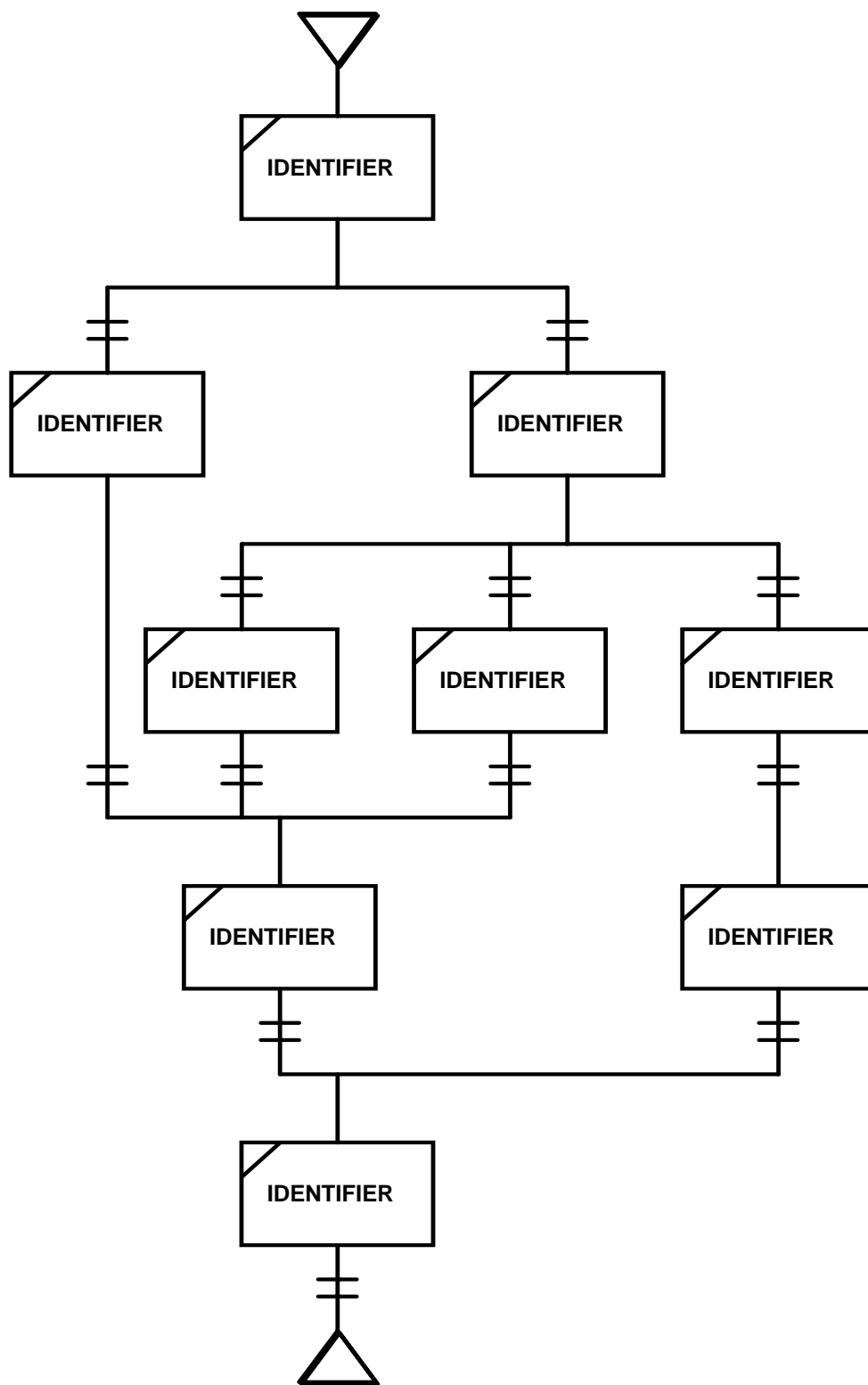


Figure 33 — Valid sequence selection diagram

Figure 34 shows an example of a valid diagram segment that shows start and end of simultaneous sequences.

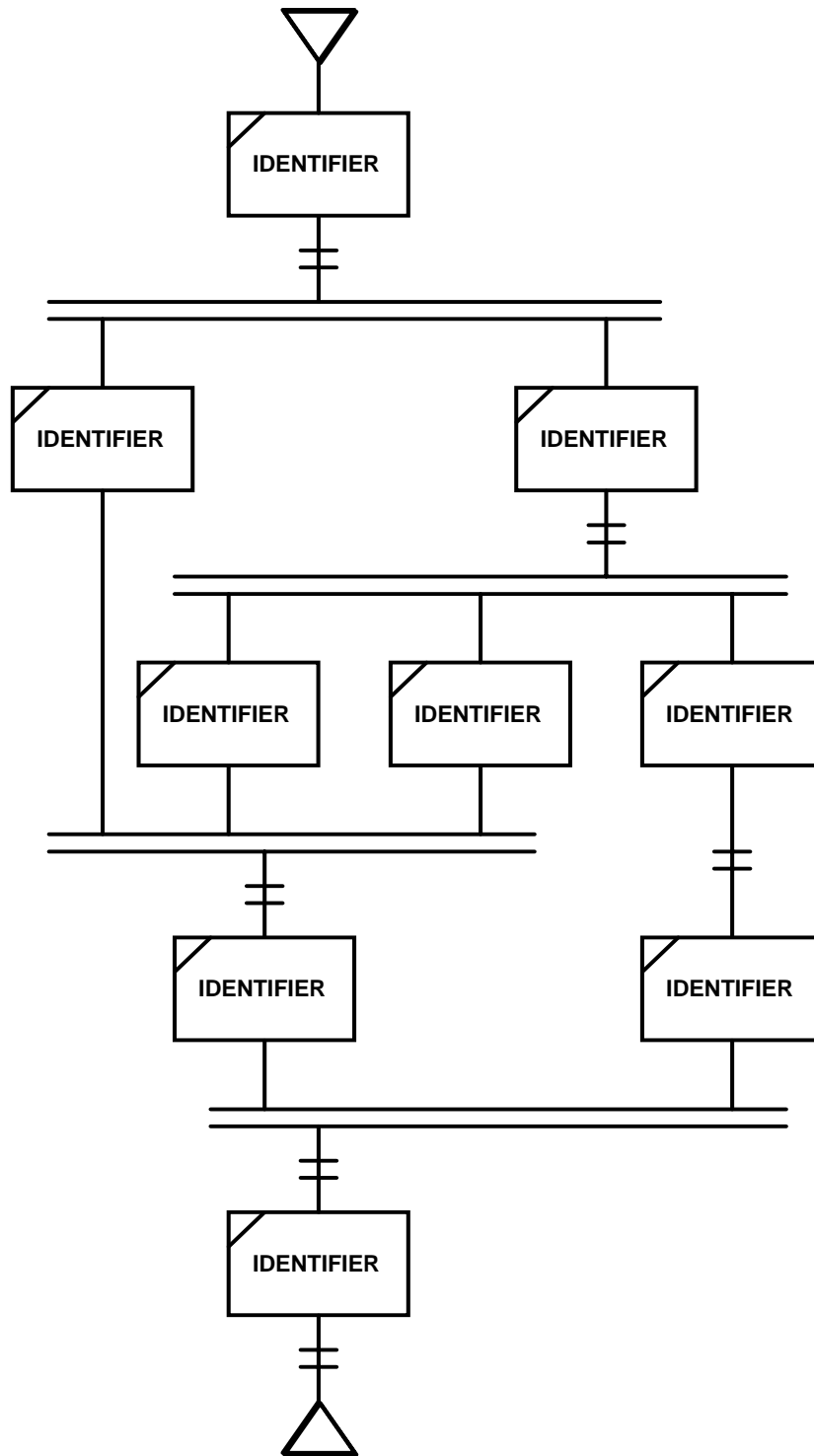


Figure 34 — Valid simultaneous sequence diagram

Looping provides for the re-execution of entities based upon transition conditions (see figure 35). This allows dynamic execution of entities based upon differing conditions.

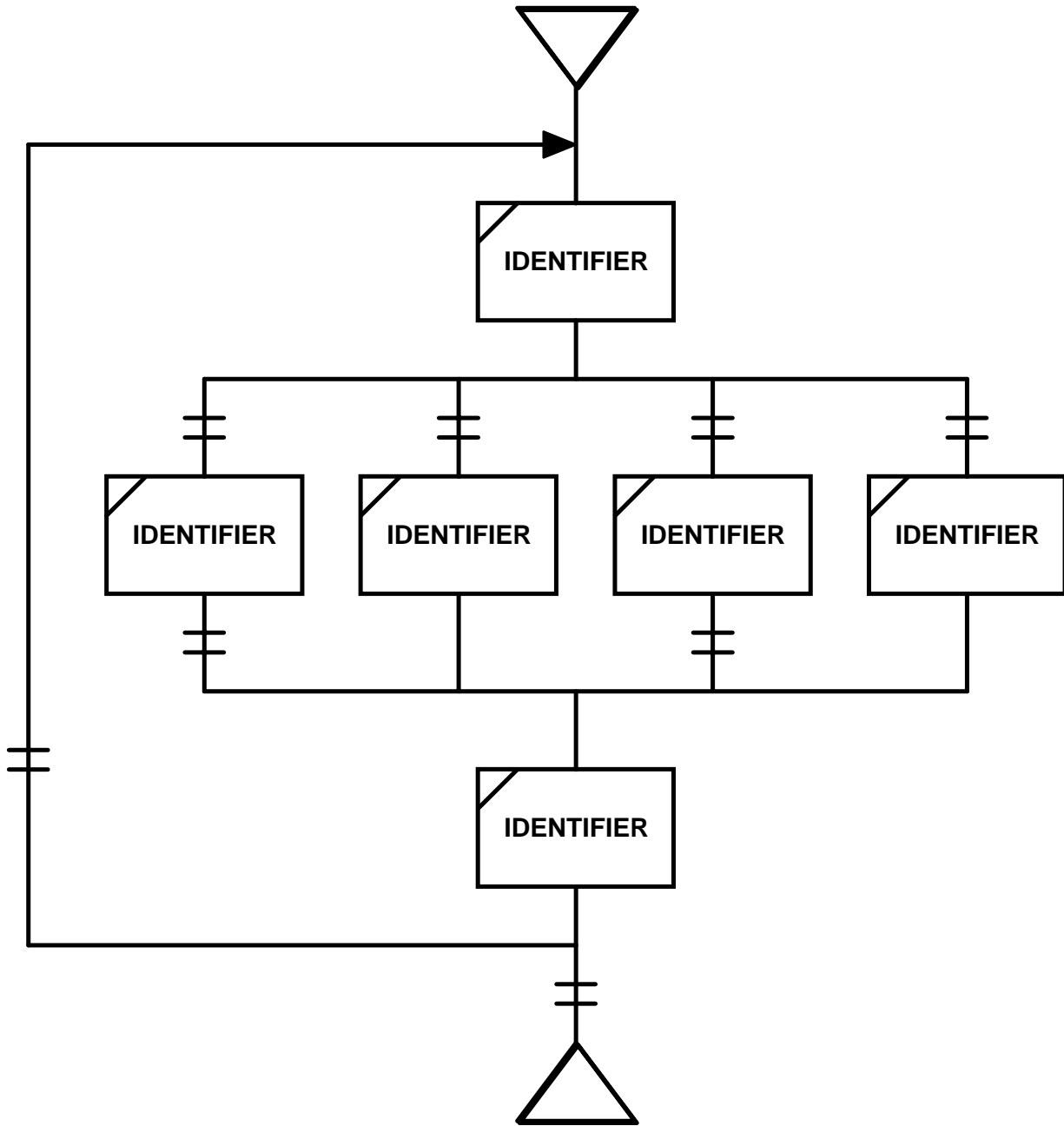


Figure 35 — Looping with explicit recipe procedural elements

This standard cannot define all valid and invalid procedure function charts. PFCs can be constructed that have unreachable procedural entities or that have an invalid execution path (e.g., in figure 36, the thread through "Phase 1" may never complete if the thread through "Phase 5" is executed).

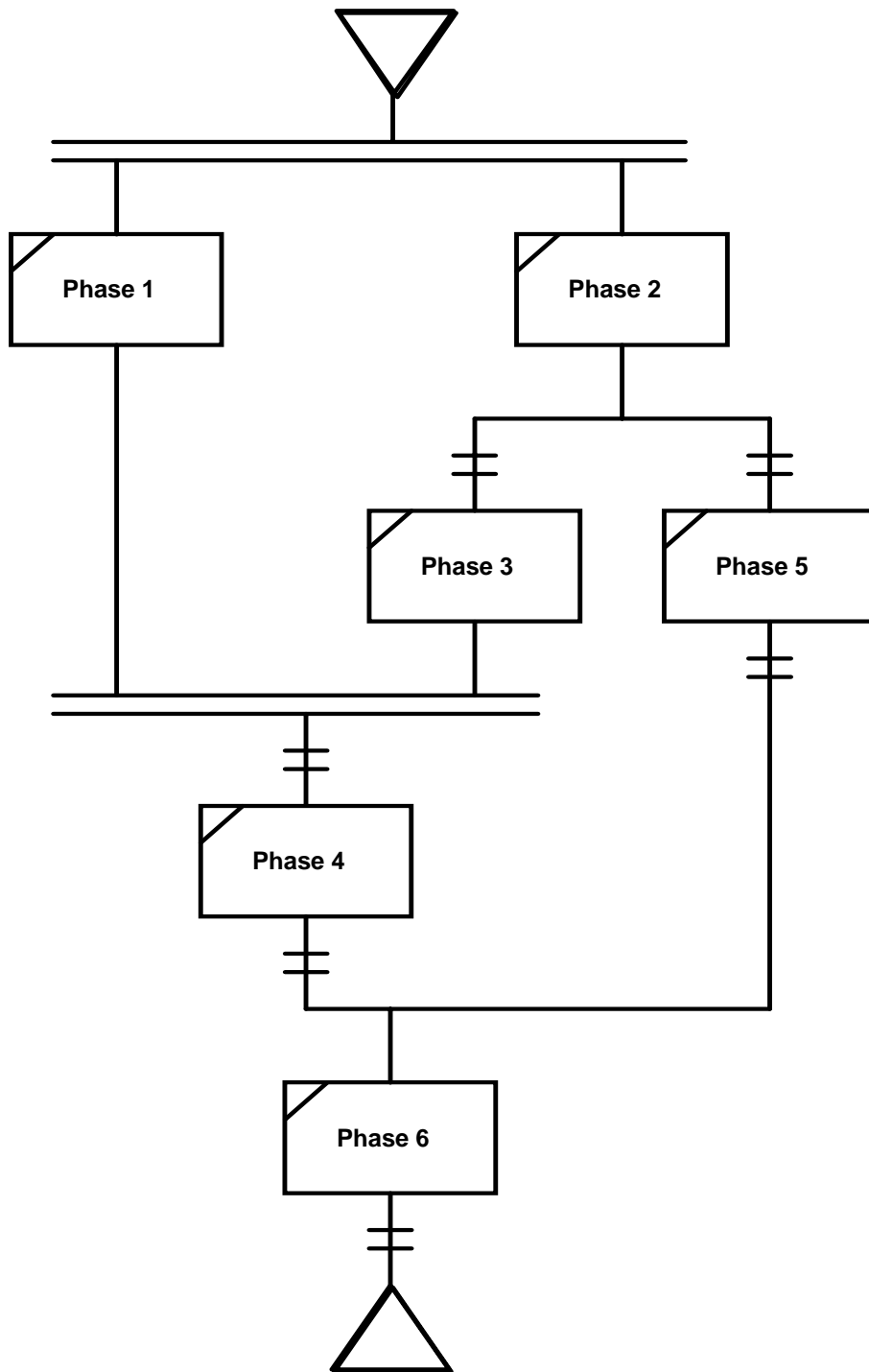


Figure 36 — Invalid procedure function chart

6.1.2 Procedure and unit procedure initiation

A depiction of the beginning of a recipe procedure is shown in figure 37. At every level below the unit procedure, directed links clearly indicate the order in which the recipe procedural elements become active or are initiated. The initiation of a recipe procedure is most likely to be related to scheduling requirements; therefore, it has a need for starting rules, some of which can be based on the schedule. A unit procedure becomes active after the transition following the allocation symbol is true.

6.1.2.1 Procedure, unit procedure, and operation completion

When the end symbol in a procedure function chart is reached, the encapsulating procedural element is complete.

6.1.2.2 Relative relationship between procedural entities

Figures 38 and 39 show two methods for depicting the relative relationship between procedural elements in procedure function charts. This relationship can be accomplished by organizing the procedural elements vertically in relation to each other. The vertical size of the procedural elements can also be varied in order to show relative relationships between depicted elements. The horizontal dashed lines show the synchronization that occurs between the operations within each unit procedure.

A compliant system shall implement at least one of the methodologies that are depicted in figures 38 and 39.

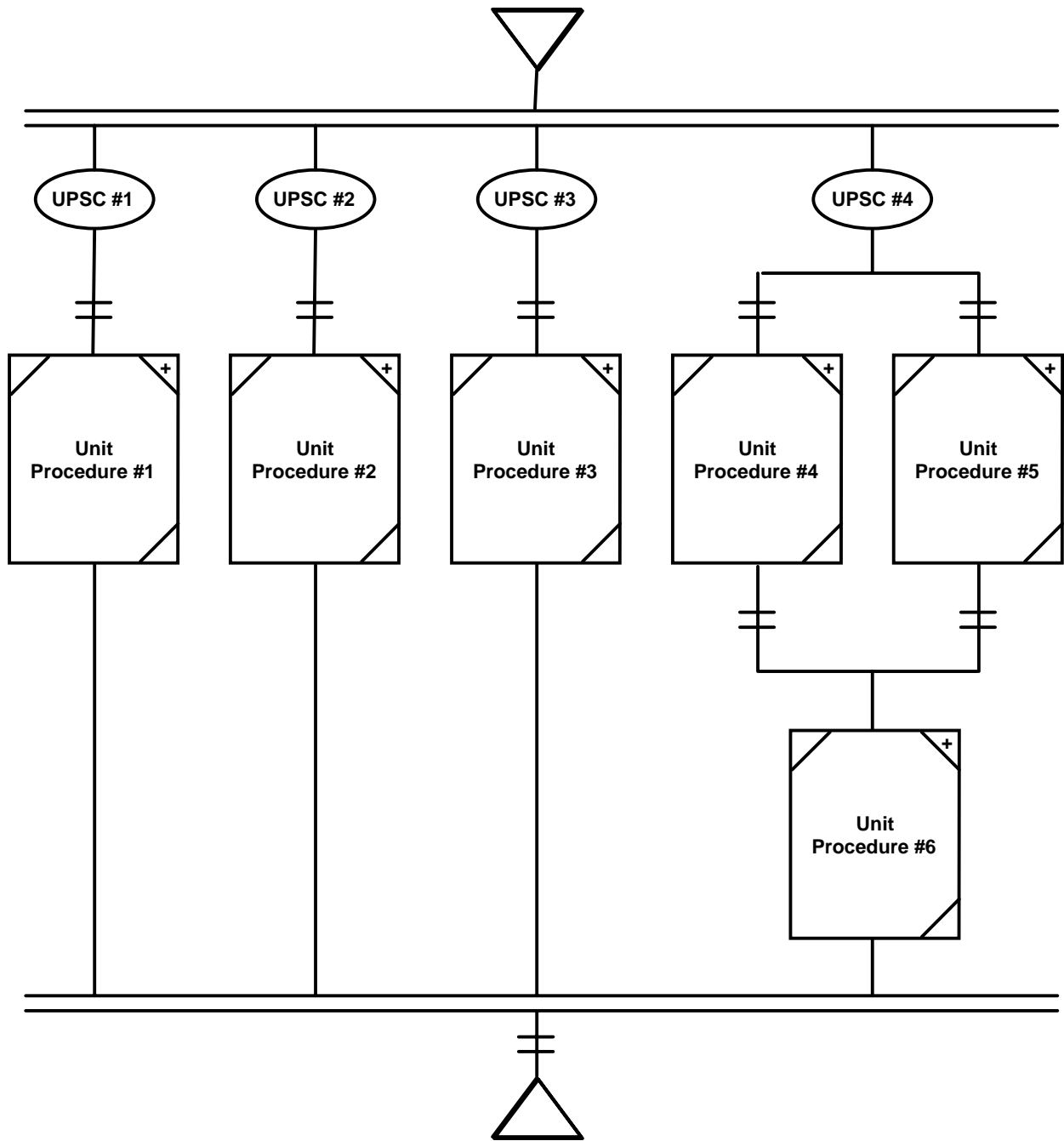


Figure 37 — Depiction of procedure and unit procedure initiation

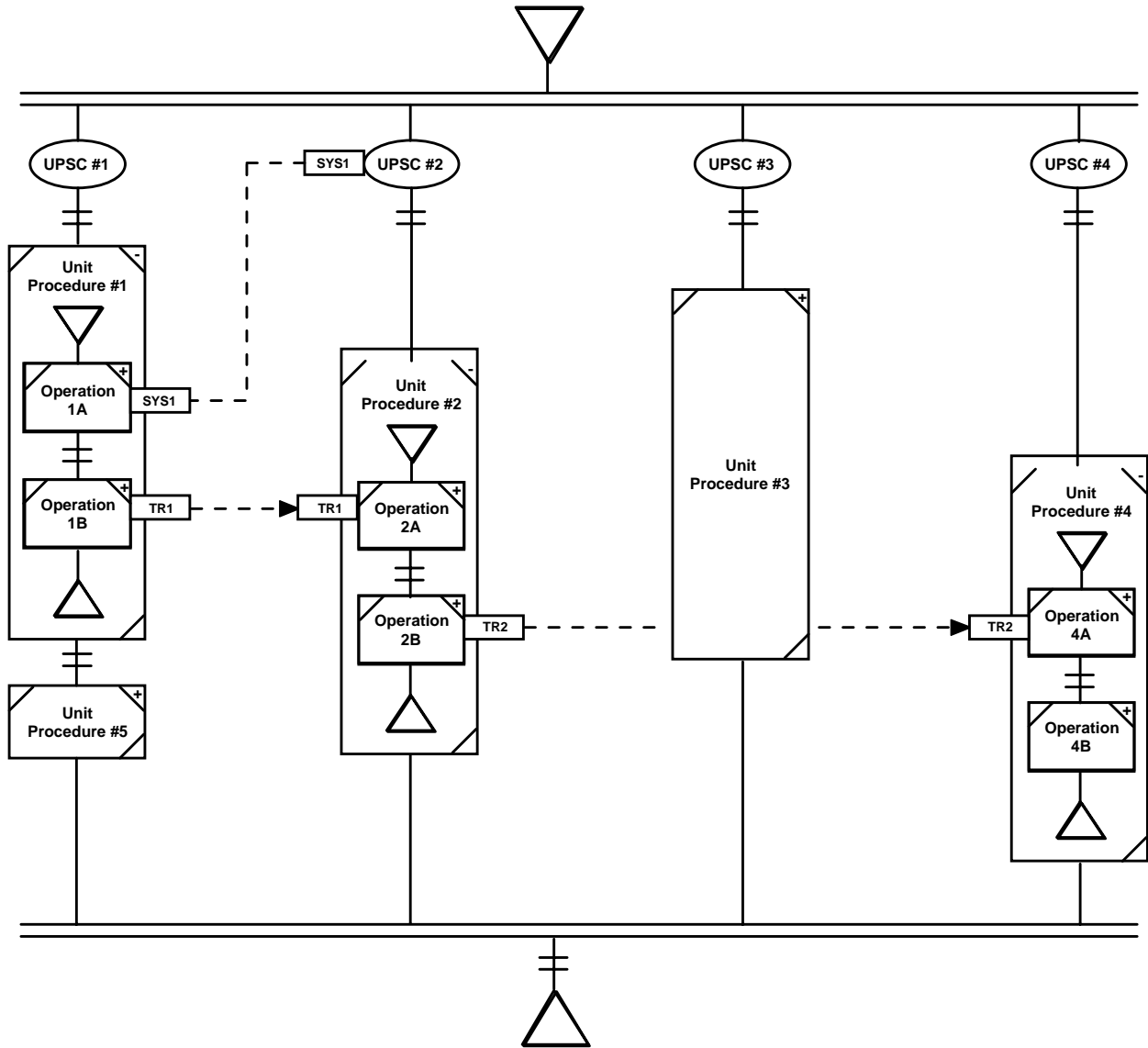


Figure 38 — Relative relationship of procedural entities

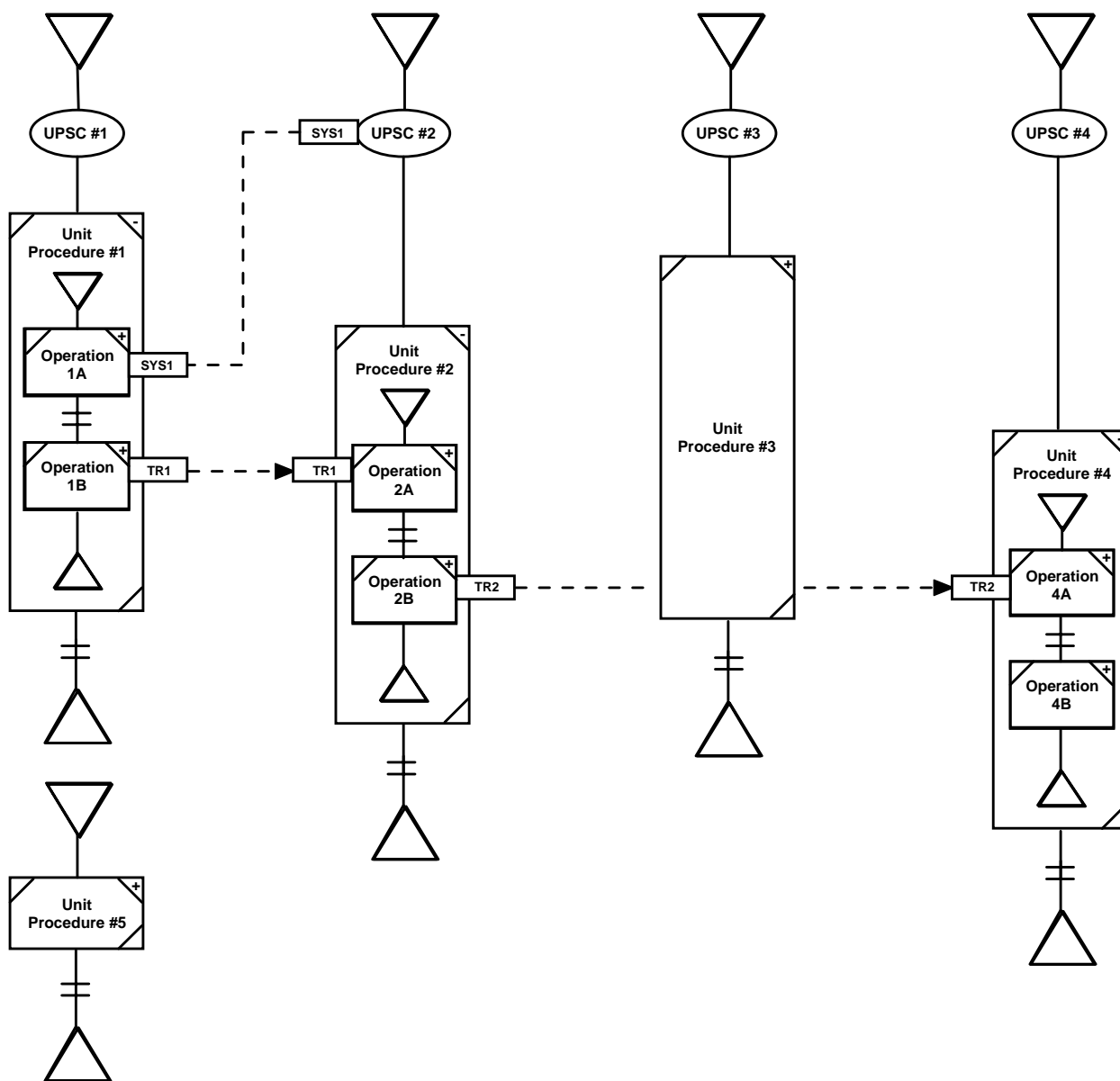


Figure 39 — Relative relationship of procedural entities – Alternate 1

When procedural elements at two different levels (e.g., unit procedures and operations) are shown on a drawing of this type, a box that shows the same grouping as the higher-level procedural element (i.e., unit procedures) shall enclose the lower-level procedural elements (i.e., operations). Unit procedure #1 in figure 38 illustrates a procedural element that is shown with a lower-level PFC that it encapsulates.

6.1.3 Non-procedural master recipe information

All other information that is part of a master recipe is related to a specific element or symbol in the master recipe procedure. This standard purposely does not specify how that relationship or reference is implemented. In a pencil and paper implementation, for example, the reference might be accomplished with something similar to a footnote or the information might be written alongside the procedural element in

question. In an electronic implementation, pop-up boxes or some other mechanism that is not yet invented might be the implementation of choice. However, the relationship shall be clearly indicated and it shall be consistent within each application.

6.1.3.1 Master recipe formula

Formula information consists of process inputs, process parameters and process outputs. The formula information shall be able to be represented in its entirety (e.g., associated with a recipe procedure), in parts (e.g., process inputs only or for a specific unit procedure) or as a summary of lower-level formula, as appropriate for the context and intended use. When depicted, the formula shall be associated with a recipe procedural element.

For example, the amount of product that a recipe produces may be associated with the recipe procedure, while the amount of material to add to a reactor may be associated with a specific phase. The use of formula permits a summation of all recipe procedural elements parameters that have been identified as process inputs, so that a list of the process inputs for an entire recipe, unit procedure, or operation can be provided.

6.1.3.2 Master recipe equipment requirements

Equipment requirements are specific to the execution of recipe procedural elements. The representation shall provide a method for the user to view the equipment requirements that are associated with each procedural element individually or for all elements in aggregate.

6.1.3.3 Header and other information

Header information and the “other information” category of recipe information may be related to the recipe in general (e.g., recipe ID, regulatory status) or to specific recipe procedural entities (e.g., protective equipment requirements, hazards of chemicals information). All header and ‘other information’ shall be able to be represented in its entirety or associated with the procedural entity to which the information is related.

6.2 Control recipe depiction

When procedure function charts are used for the depiction of a control recipe, they shall follow the same principles that are defined for the use of procedure function charts for the depiction of a master recipe. In this case, however, the depiction of the control recipe will also, usually, become an active display in automated systems. Because the relationship between recipe procedural elements and equipment procedural elements is known during the execution of the control recipe, colors and/or other means of demarcation may be used to indicate the status of the procedural elements.

6.3 Exception handling

Special processing may be included in the recipe in order to handle product-specific exceptions, in addition to any equipment logic that is used for handling equipment-related exceptions. These product-specific exceptions are generally related to the method of manufacturing a product, often its quality, and they are not specific to equipment. They are interwoven into the normal procedure; therefore, they are not easily distinguished from normal processing and they only become active in the event of an exception.

The extent of the effect of exception handling in the recipe procedure is normally confined to unit procedures because the units operate independently. In some cases, however, commands, states, and/or modes are propagated to another unit when there is a common concern (e.g., a transfer or concurrent processing with common time constraints). In some circumstances, an entire recipe may have to be commanded to a specified state and/or mode.

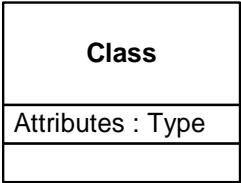

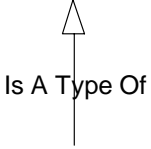


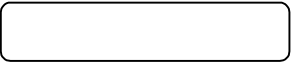
This page intentionally left blank.

Annex A (normative) — Data modeling technique

A.1 UML notation

Table A.1 defines the UML notation that is used in this standard (see J. Rumbaugh/I. Jacobson, The Unified Modeling Language Reference Manual, 1999, Addison Wesley Longman, Inc.).

Table A.1 — UML notation

SYMBOL	DEFINITION
	<p>Defines a class of objects, each with the same types of attributes. Each object is uniquely identifiable or enumerable. No operations or methods are listed for the classes. Attributes with a '-' before their name indicate attributes that are generally optional in any use of the class.</p>
	<p>An association between elements of a class and elements of another or the same class. Each association is identified. Can have the expected number or range of members of the subclass, where 'n' indicates an indeterminate number (e.g., 0,n means that zero or more members of the subclass may exist).</p>
	<p>Generalization (arrow points to the super class) shows that an element of the class is a specialized type of the super class.</p>
	<p>Dependence (i.e., tightly bound relationship between the items) shows that an element of the class depends on an element of another class.</p>
	<p>Aggregation (i.e., made up of) shows that an element of the class is made up of elements of other classes.</p>
	<p>A class of object that is an instance of another class of object.</p>

A.2 Definitions

A.2.1 class:

a description of a set of objects that share the same attributes, behaviors, relationships, and semantics.

A.2.2 encapsulation:

a technique that separates the external aspects of an object from the internal, implementation details of the object (also called information hiding).

A.2.3 instance:

a term that is used to refer to an object that belongs to a particular class but that is not itself a class or a subclass. For example, "reactor401" is an instance of the class "reactor".

A.2.4 model:

a formal abstract representation of a system. A model is usually presented as a collection of diagrams and a data dictionary.

A.2.5 object:

an entity that is composed of state and behavior. State is the value of all attributes at a given time. An attribute is a piece of information that qualifies the object. The behavior of an object is the functionality that is contained in the object that is necessary to manipulate the attributes.

A.2.6 subclass:

a class that is a special case of a more general class (e.g., glass-lined reactor is a subclass of reactor class).

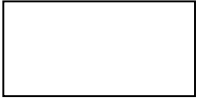
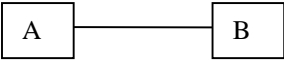
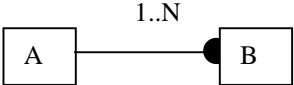
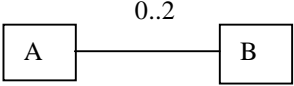
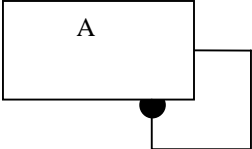
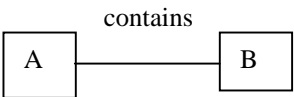
A.2.7 unified modeling language (UML):

a language that is used for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.

A.3 ERD notation

Table A.2 defines the ERD notation that is used in this standard.

Table A.2 — ERD notation

SYMBOL	DEFINITION
	<p>Defines an entity.</p>
	<p>For each occurrence of A, there is one and only one occurrence of B. The association at B may also be marked with the number 1.</p>
	<p>Numerically specified association. In this example, for each occurrence of A, there may be one or more occurrences of B. Another example is 0..N. If no numeric association is given, 0..N is assumed.</p>
	<p>Numerically specified association: 0 to some positive number. In this example, for each occurrence of A, there may be 0 to 2 occurrences of B.</p>
	<p>Looped association. An occurrence of A may be made up of zero or more occurrences of entities of the same type. An optional association is that an occurrence of entity A may contain zero or more occurrences of entities of the same type. Another usage is that an occurrence of entity A may contain zero or more occurrences of entities of the same type.</p>
	<p>The association between entities is labeled in order to specify the nature of the relationship. The label applies to the entity that it is closest to. In this example, the association is read as follows: Each occurrence of A <i>contains</i> one and only one occurrence of B.</p>

This page intentionally left blank.

Annex B (normative) — SQL definition listing

ISA hereby grants a non-exclusive, royalty-free, limited license under ISA's copyright in the standard, to copy, display and distribute this section of this ISA standard (including software included in or defined by such section), as follows:

1. Producers of products or services intended to comply with the standard may incorporate this designated section, but only to the extent reasonably necessary to make, use, and distribute any product or service (including product documentation) that is compliant with this standard.
2. End users of a product or service made by a producer acting under the preceding license may reproduce and use the designated section, but only to the extent reasonably necessary to enjoy the intended functions of the product or service and to maintain, configure, or reconfigure systems to be compliant.
3. Persons providing education on or promotion of the standard may copy, display and distribute the designated section, but only to the extent reasonably necessary to provide information related to the standard.

Except as expressly permitted, all other reproduction and distribution without permission of ISA is prohibited. All copies of this section of the standard made or distributed under this license must cite the standard and include the following notice of copyright:

Copyright © 2001 by ISA–The Instrumentation, Systems, and Automation Society. All rights reserved. Used with permission of ISA.

This annex contains the ISO/IEC9075:1992 standard definition of all of the tables that are defined in clause 5.

```
CREATE TABLE BXT_Exchange (
  ExchangeID      CHAR (32)      NOT NULL,
  ExchangeValue   CHAR (128)     NOT NULL,
  PRIMARY KEY (ExchangeID))

INSERT INTO BXT_Exchange (ExchangeID, ExchangeValue)
VALUES ('Schema','ANSI/ISA-88.00.02-2001')
VALUES ('Delimiter','/')
VALUES ('ToolID','ToolName')
VALUES ('ToolVersion','4.0')
VALUES ('ToolSchema','1.2')

CREATE TABLE BXT_EnumerationSet (
  EnumSet        CHAR (32)      NOT NULL,
  Description     CHAR (255),
  PRIMARY KEY (EnumSet))

CREATE TABLE BXT_Enumeration (
  EnumSet        CHAR (32)      NOT NULL,
  EnumValue      INTEGER        NOT NULL,
  EnumString     CHAR (32),
  Description     CHAR (255),
  PRIMARY KEY (EnumSet, EnumValue))
```

```
INSERT INTO BXT_EnumerationSet (EnumSet, Description)
VALUES ('Boolean',
       'Defines a set of Boolean values')
VALUES ('DirectionType',
       'Defines how a parameter is intended to be handled')
VALUES ('EquipmentLevel',
       'Defines the equipment hierarchical level for equipment elements')
VALUES ('EquipmentType',
       'Defines the type of equipment record for equipment elements')
VALUES ('EvaluationRule',
       'Defines the evaluation rules for equipment properties')
VALUES ('FormulaSubType',
       'Defines the recipe formula types')
VALUES ('FormulaType',
       'User supplied formula sub type definitions')
VALUES ('LinkDepiction',
       'Defines how links between recipe elements are to be depicted')
VALUES ('LinkToType',
       'Defines if a link references a step or a transition')
VALUES ('LinkType',
       'Defines the type of link')
VALUES ('RE_Type',
       'Defines the recipe element, either recipe procedure level or allocation entity')
VALUES ('RE_Use',
       'Defines how a recipe element is used in a recipe')
VALUES ('RecipeStatus',
       'Defines the possible status of a recipe')
VALUES ('RecordSet',
       'Defines the enumeration set used to classify a record into a category of batch history
       information.')
VALUES ('RecordSetControlRecipe',
       'Provides further history record classification under the category of ControlRecipe.')
VALUES ('RecordSetMasterRecipe',
       'Provides further history record classification under the category of MasterRecipe.')
VALUES ('RecordSetExecutionInfo',
       'Provides further history record classification under the category of ExecutionInfo.')
VALUES ('RecordSetMaterialInfo',
       'Provides further history record classification under the category of MaterialInfo.')
VALUES ('RecordSetContinuousData',
       'Provides further history record classification under the category of ContinuousData.')
VALUES ('RecordSetEvents',
       'Provides further history record classification under the category of Events.')
VALUES ('RecordSetOperatorChange',
       'Provides further history record classification under the category of OperatorChange.')
VALUES ('RecordSetOperatorComment',
       'Provides further history record classification under the category of OperatorComment.')
VALUES ('RecordSetAnalysisData',
       'Provides further history record classification under the category of AnalysisData.')
VALUES ('RecordSetLateRecord',
       'Provides further history record classification under the category of LateRecord.')
VALUES ('RecordSetRecipeData',
       'Provides further history record classification under the category of RecipeData.')
VALUES ('RecordSetRecipeSpecified',
       'Provides further history record classification under the category of RecipeSpecified.')
```

VALUES ('RecordSetSummaryData',
 'Provides further history record classification under the category of SummaryData.')

VALUES ('ScheduleAction',
 'Defines the intended action of the schedule entry')

VALUES ('ScheduleMode',
 'Defines the mode which the schedule entry begins execution in')

VALUES ('ScheduleStatus',
 'Defines the possible status of a schedule')

VALUES ('SE_Type',
 'Defines the type of entity in a schedule record')

VALUES ('ValueDataType',
 'Defines how a value is represented (e.g. Boolean, float, etc.)')

VALUES ('ValueType',
 'Defines how a value string is interpreted')

INSERT INTO BXT_Enumeration (EnumSet, EnumValue, EnumString, Description)

VALUES ('Boolean', 0, 'FALSE',
 'Defines a Boolean value')

VALUES ('Boolean', 1, 'TRUE', '')

VALUES ('DirectionType', 0, 'Invalid', 'Entry not valid')

VALUES ('DirectionType', 1, 'Internal',
 'Identifies how a parameter is handled.')

VALUES ('DirectionType', 2, 'Input',
 'The Recipe Element receives the value from an external source.')

VALUES ('DirectionType', 3, 'Output',
 'The Recipe Element creates the value and makes it available for external use.')

VALUES ('DirectionType', 4, 'Input/Output',
 'The Recipe Element and external element exchange the value, and may change its value.')

VALUES ('EquipmentLevel', 0, 'Invalid', 'Entry not valid')

VALUES ('EquipmentLevel', 1, 'Enterprise',
 'Identifies the equipment hierarchical level for BXT_EquipElement')

VALUES ('EquipmentLevel', 2, 'Site', '')

VALUES ('EquipmentLevel', 3, 'Area', '')

VALUES ('EquipmentLevel', 4, 'Process Cell', '')

VALUES ('EquipmentLevel', 5, 'Unit', '')

VALUES ('EquipmentLevel', 6, 'Equipment Module', '')

VALUES ('EquipmentLevel', 7, 'Control Module', '')

VALUES ('EquipmentType', 0, 'Invalid', 'Entry not valid')

VALUES ('EquipmentType', 1, 'Class',
 'Identifies the record type for BXT_EquipElement')

VALUES ('EquipmentType', 2, 'Element', '')

VALUES ('EvaluationRule', 0, 'Invalid', 'Entry not valid')

VALUES ('EvaluationRule', 1, '=',
 'Equals comparison operator for equipment properties')

VALUES ('EvaluationRule', 2, '<>',
 'Not equals comparison operator for equipment properties')

VALUES ('EvaluationRule', 3, '<',
 'Less than comparison operator for equipment properties')

VALUES ('EvaluationRule', 4, '>',
 'Greater than comparison operator for equipment properties')

VALUES ('EvaluationRule', 5, '<=',
 'Less than or equals comparison operator for equipment properties')

VALUES ('EvaluationRule', 6, '>=',
 'Greater than or equals comparison operator for equipment properties')
 VALUES ('EvaluationRule', 7, 'Member',
 'Is a member of comparison operator for equipment properties')
 VALUES ('EvaluationRule', 8, 'Not member',
 'Is not a member of comparison operator for equipment properties')
 VALUES ('EvaluationRule', 9, 'Not',
 'Not comparison operator for equipment properties')
 VALUES ('FormulaType', 0, 'Invalid', 'Entry not valid')
 VALUES ('FormulaType', 1, 'Process Input', 'Recipe Formula type')
 VALUES ('FormulaType', 2, 'Process Output', '')
 VALUES ('FormulaType', 3, 'Process Parameter', '')
 VALUES ('FormulaSubType', 0, 'Invalid', 'Entry not valid')
 VALUES ('LinkDepiction', 0, 'Invalid', 'Entry not valid')
 VALUES ('LinkDepiction', 1, 'None', 'No link depiction')
 VALUES ('LinkDepiction', 2, 'Line', 'Link shown with line only')
 VALUES ('LinkDepiction', 3, 'ID', 'Link shown with identifier only')
 VALUES ('LinkDepiction', 4, 'Line & ID',
 'Link shown with line and identification')
 VALUES ('LinkDepiction', 5, 'Line & Arrow',
 'Link shown with line and material flow arrow')
 VALUES ('LinkDepiction', 6, 'Line, Arrow, & ID',
 'Link shown with line, material flow arrow and identification')
 VALUES ('LinkToType', 0, 'Invalid', 'Entry not valid')
 VALUES ('LinkToType', 1, 'Recipe Element',
 'Link is referencing an entry in the BXT_MRecipeElement table')
 VALUES ('LinkToType', 2, 'Transition',
 'Link is referencing an entry in the BXT_MRecipeTransition table')
 VALUES ('LinkType', 0, 'Invalid', 'Entry not valid')
 VALUES ('LinkType', 1, 'ControlLink',
 'Defines a link between recipe elements that indicates a flow of procedural control.')
 VALUES ('LinkType', 2, 'TransferLink',
 'Defines a link between recipe elements that indicates a material transfer.')
 VALUES ('LinkType', 3, 'SynchronizationLink',
 'Defines a link between recipe elements where there is some form of synchronization.')
 VALUES ('RE_Type', 0, 'Invalid', 'Entry not valid')
 VALUES ('RE_Type', 1, 'Master Recipe',
 'Specifies the type of recipe element.')
 VALUES ('RE_Type', 2, 'Procedure', '')
 VALUES ('RE_Type', 3, 'Unit Procedure', '')
 VALUES ('RE_Type', 4, 'Operation', '')
 VALUES ('RE_Type', 5, 'Phase', '')
 VALUES ('RE_Type', 6, 'Allocation', '')
 VALUES ('RE_Type', 7, 'Begin', '')
 VALUES ('RE_Type', 8, 'End', '')
 VALUES ('RE_Type', 9, 'Start Parallel', '')
 VALUES ('RE_Type', 10, 'End Parallel', '')
 VALUES ('RE_Type', 11, 'Start Branch', '')
 VALUES ('RE_Type', 12, 'End Branch', '')
 VALUES ('RE_Use', 0, 'Invalid', 'Entry not valid')
 VALUES ('RE_Use', 1, 'Linked',
 'A recipe element (RE) may have several referencing RE Steps')
 VALUES ('RE_Use', 2, 'Embedded',
 'An RE has only one referencing RE, one RE is defined for each use of the RE.')

VALUES ('RE_Use', 3, 'Copied',
 'The same as Embedded, but the specific RE was modified from its original definition.')

VALUES ('RecipeStatus', 0, 'Invalid', 'Entry not valid')

VALUES ('RecipeStatus', 1, 'Approved for Production',
 'Recipe was approved for production.')

VALUES ('RecipeStatus', 2, 'Approved for Test',
 'Recipe was only approved for test.')

VALUES ('RecipeStatus', 3, 'Not Approved',
 'Recipe was not approved for production or test.')

VALUES ('RecipeStatus', 4, 'Inactive', 'Recipe was not active.')

VALUES ('RecipeStatus', 5, 'Obsolete', 'Recipe was obsolete.')

VALUES ('RecordSet', 0, 'Invalid', 'Entry not valid')

VALUES ('RecordSet', 1, 'RecordSetControlRecipe',
 'Defines that a batch history information record is part of the ControlRecipe category.')

VALUES ('RecordSet', 2, 'RecordSetMasterRecipe', '')

VALUES ('RecordSet', 3, 'RecordSetExecutionInfo', '')

VALUES ('RecordSet', 4, 'RecordSetMaterialInfo', '')

VALUES ('RecordSet', 5, 'RecordSetContinuousData', '')

VALUES ('RecordSet', 6, 'RecordSetEvents', '')

VALUES ('RecordSet', 7, 'RecordSetOperatorChange', '')

VALUES ('RecordSet', 8, 'RecordSetOperatorComment', '')

VALUES ('RecordSet', 9, 'RecordSetAnalysisData', '')

VALUES ('RecordSet', 10, 'RecordSetLateRecord', '')

VALUES ('RecordSet', 11, 'RecordSetRecipeData', '')

VALUES ('RecordSet', 12, 'RecordSetRecipeSpecified', '')

VALUES ('RecordSet', 13, 'RecordSetSummaryData', '')

VALUES ('RecordSetControlRecipe', 0, 'Invalid', 'Entry not valid')

VALUES ('RecordSetControlRecipe', 1, 'Entire Control Recipe',
 'History record is related to the entire control recipe.')

VALUES ('RecordSetMasterRecipe', 0, 'Invalid', 'Entry not valid')

VALUES ('RecordSetMasterRecipe', 1, 'Entire Master Recipe',
 'History record is related to the entire master recipe.')

VALUES ('RecordSetExecutionInfo', 0, 'Invalid', 'Entry not valid')

VALUES ('RecordSetExecutionInfo', 1, 'Allocation', '')

VALUES ('RecordSetExecutionInfo', 2, 'De-allocation', '')

VALUES ('RecordSetExecutionInfo', 3, 'State Change', '')

VALUES ('RecordSetExecutionInfo', 4, 'State Command', '')

VALUES ('RecordSetExecutionInfo', 5, 'Mode Change', '')

VALUES ('RecordSetExecutionInfo', 6, 'Mode Command', '')

VALUES ('RecordSetExecutionInfo', 7, 'Procedural Entity Message', '')

VALUES ('RecordSetExecutionInfo', 8, 'Procedural Entity Alarm', '')

VALUES ('RecordSetExecutionInfo', 9, 'Procedural Entity Version', '')

VALUES ('RecordSetExecutionInfo', 10, 'Procedural Entity Prompt', '')

VALUES ('RecordSetExecutionInfo', 11, 'Procedural Entity Prompt Resp', '')

VALUES ('RecordSetMaterialInfo', 0, 'Invalid', 'Entry not valid')

VALUES ('RecordSetMaterialInfo', 1, 'Material Consumption', '')

VALUES ('RecordSetMaterialInfo', 2, 'Material Production', '')

VALUES ('RecordSetMaterialInfo', 3, 'Material Allocation', '')

VALUES ('RecordSetMaterialInfo', 4, 'Material De-allocation', '')

VALUES ('RecordSetContinuousData', 0, 'Invalid', 'Entry not valid')

VALUES ('RecordSetContinuousData', 1, 'Continuous Data Value', '')

VALUES ('RecordSetContinuousData', 2, 'Trend Association', '')

VALUES ('RecordSetContinuousData', 3, 'Trend Disassociation', '')

VALUES ('RecordSetEvents', 0, 'Invalid', 'Entry not valid')

VALUES ('RecordSetEvents', 1, 'General Event', ")
 VALUES ('RecordSetOperatorChange', 0, 'Invalid', 'Entry not valid')
 VALUES ('RecordSetOperatorChange', 1, 'General Operator Intervention', ")
 VALUES ('RecordSetOperatorComment', 0, 'Invalid', 'Entry not valid')
 VALUES ('RecordSetOperatorComment', 1, 'General Operator Comment', ")
 VALUES ('RecordSetAnalysisData', 0, 'Invalid', 'Entry not valid')
 VALUES ('RecordSetAnalysisData', 1, 'General Analysis Message', ")
 VALUES ('RecordSetLateRecord', 0, 'Invalid', 'Entry not valid')
 VALUES ('RecordSetLateRecord', 1, 'General Late Record', ")
 VALUES ('RecordSetRecipeData', 0, 'Invalid', 'Entry not valid')
 VALUES ('RecordSetRecipeData', 1, 'Generic Recipe Data', ")
 VALUES ('RecordSetRecipeData', 2, 'Recipe Parameter Value Change', ")
 VALUES ('RecordSetRecipeData', 3, 'Recipe Result Data', ")
 VALUES ('RecordSetRecipeSpecified', 0, 'Invalid', 'Entry not valid')
 VALUES ('RecordSetRecipeSpecified', 1, 'Generic Recipe Specified Data', ")
 VALUES ('RecordSetSummaryData', 0, 'Invalid', 'Entry not valid')
 VALUES ('RecordSetSummaryData', 1, 'Generic Summary Data', ")
 VALUES ('RecordSetSummaryData', 2, 'Utilities Consumption', ")
 VALUES ('RecordSetSummaryData', 3, 'Equipment Run Time', ")
 VALUES ('ScheduleChange', 0, 'Invalid', 'Entry not valid')
 VALUES ('ScheduleChange', 1, 'New', 'Schedule record change action')
 VALUES ('ScheduleChange', 2, 'Update', ")
 VALUES ('ScheduleChange', 3, 'Delete', ")
 VALUES ('ScheduleMode', 0, 'Invalid', 'Entry not valid')
 VALUES ('ScheduleMode', 1, 'Automatic', 'Schedule record mode')
 VALUES ('ScheduleMode', 2, 'Semi-Automatic', ")
 VALUES ('ScheduleMode', 3, 'Manual', ")
 VALUES ('ScheduleMode', 4, 'Not Specified', ")
 VALUES ('ScheduleStatus', 0, 'Invalid', 'Entry not valid')
 VALUES ('ScheduleStatus', 1, 'Complete', 'Batch schedule record status')
 VALUES ('ScheduleStatus', 2, 'In-progress', ")
 VALUES ('ScheduleStatus', 3, 'Scheduled', ")
 VALUES ('ScheduleStatus', 4, 'Schedule Hold', ")
 VALUES ('ScheduleStatus', 5, 'Not Specified', ")
 VALUES ('SE_Type', 0, 'Invalid', 'Entry not valid')
 VALUES ('SE_Type', 1, 'Campaign',
 'Defines the type of Scheduled Entry')
 VALUES ('SE_Type', 2, 'Batch', ")
 VALUES ('SE_Type', 3, 'Unit Procedure', ")
 VALUES ('SE_Type', 4, 'Operation', ")
 VALUES ('SE_Type', 5, 'Phase', ")
 VALUES ('ValueDataType', 0, 'Invalid', 'Entry not valid')
 VALUES ('ValueDataType', 1, 'Boolean',
 'Defines the data type that is expected for an associated value. ')
 VALUES ('ValueDataType', 2, '8-Bit string', ")
 VALUES ('ValueDataType', 3, '16-Bit string', ")
 VALUES ('ValueDataType', 4, '32-Bit string', ")
 VALUES ('ValueDataType', 5, '8-Bit unsigned integer', ")
 VALUES ('ValueDataType', 6, '16-Bit unsigned integer', ")
 VALUES ('ValueDataType', 7, '32-Bit unsigned integer', ")
 VALUES ('ValueDataType', 8, '8-Bit signed integer', ")
 VALUES ('ValueDataType', 9, '16-Bit signed integer', ")
 VALUES ('ValueDataType', 10, '32-Bit signed integer', ")
 VALUES ('ValueDataType', 11, '32-Bit float', ")

VALUES ('ValueDataType', 12, 'Double float', '')
 VALUES ('ValueDataType', 13, 'Octet string', '')
 VALUES ('ValueDataType', 14, 'DateTime', '')
 VALUES ('ValueType', 0, 'Invalid', 'Entry not valid')
 VALUES ('ValueType', 1, 'Constant',
 'Defines how a value string is interpreted. It contains a fixed value as a string. ')
 VALUES ('ValueType', 2, 'Reference',
 'Defines how a value string is interpreted. It points to the source of the value. ')
 VALUES ('ValueType', 3, 'Equation',
 'Defines that a value string is interpreted as an expression to be evaluated in order to determine
 the value. ')
 VALUES ('ValueType', 4, 'External',
 'Value is supplied by some external means, and it is not contained in the recipe (i.e., value may
 be supplied by an operator entry or by a scheduling system).')

```
CREATE TABLE BXT_MRecipeElement (
  RE_ID          CHAR (128)      NOT NULL,
  REVersion      CHAR (16)       NOT NULL,
  VersionDate    DATETIME,
  ApprovalDate   DATETIME,
  EffectiveDate  DATETIME,
  ExpirationDate DATETIME,
  Author         CHAR (32),
  ApprovedBy     CHAR (32),
  ProcessCellID CHAR (32),
  ProductID      CHAR (32),
  UsageConstraint CHAR (255),
  Description     CHAR (255),
  Status         INTEGER,
  RE_Type        INTEGER,
  RE_Function    CHAR (255),
  RE_Use         INTEGER,
  DerivedRE      CHAR (128),
  DerivedVersion CHAR (16),
  PRIMARY KEY (RE_ID, REVersion))

CREATE TABLE BXT_MRecipeStep (
  ParentRE      CHAR (128)      NOT NULL,
  ParentVersion CHAR (16)       NOT NULL,
  StepID        CHAR (128)      NOT NULL,
  RE_ID         CHAR (128)      NOT NULL,
  REVersion     CHAR (16)       NOT NULL,
  VerticalStart FLOAT,
  VerticalStop  FLOAT,
  HorizontalStart FLOAT,
  HorizontalStop FLOAT,
  ScaleReference FLOAT,
  ScaleEngrUnits CHAR (32),
  MaximumScale  FLOAT,
  MinimumScale  FLOAT,
  PRIMARY KEY (ParentRE, ParentVersion, StepID),
  FOREIGN KEY (RE_ID, REVersion)
  REFERENCES BXT_MRecipeElement (RE_ID, REVersion))
```

```
CREATE TABLE BXT_MRecipeTransition (  
  RE_ID          CHAR (128) NOT NULL,  
  REVersion      CHAR (16)  NOT NULL,  
  TransitionID   CHAR (128) NOT NULL,  
  Condition      CHAR (255),  
  VerticalStart  FLOAT,  
  VerticalStop   FLOAT,  
  HorizontalStart  FLOAT,  
  HorizontalStop  FLOAT,  
  PRIMARY KEY (RE_ID, REVersion, TransitionID),  
  FOREIGN KEY (RE_ID, REVersion)  
    REFERENCES BXT_MRecipeElement (RE_ID, REVersion))
```

```
CREATE TABLE BXT_MRecipeLink (  
  RE_ID          CHAR (128) NOT NULL,  
  REVersion      CHAR (16)  NOT NULL,  
  LinkID         CHAR (32)  NOT NULL,  
  FromType       INTEGER,  
  FromElement    CHAR (128),  
  ToType         INTEGER,  
  ToElement      CHAR (128),  
  LinkType       INTEGER,  
  VerticalStart  FLOAT,  
  VerticalStop   FLOAT,  
  HorizontalStart  FLOAT,  
  HorizontalStop  FLOAT,  
  Depiction      INTEGER,  
  EvaluationOrder  INTEGER,  
  PRIMARY KEY (RE_ID, REVersion, LinkID),  
  FOREIGN KEY (RE_ID, REVersion)  
    REFERENCES BXT_MRecipeElement (RE_ID, REVersion) )
```

```
CREATE TABLE BXT_MRecipeElementParameter (  
  RE_ID          CHAR (128) NOT NULL,  
  REVersion      CHAR (16)  NOT NULL,  
  ParameterID    CHAR(32)   NOT NULL,  
  ParentParamID  CHAR(32),  
  DataInterpretation  INTEGER,  
  DataDirection  INTEGER,  
  DefaultValue   CHAR(128),  
  Description     CHAR(255),  
  EngrUnits      CHAR(32),  
  EnumSet        CHAR (32),  
  DefaultScaling  INTEGER,  
  ParamType      INTEGER,  
  ParamSubType   INTEGER,  
  PRIMARY KEY (RE_ID, REVersion, ParameterID),  
  FOREIGN KEY (RE_ID, REVersion)  
    REFERENCES BXT_MRecipeElement (RE_ID, REVersion))
```



```

CREATE TABLE BXT_MRecipeStepParameter (
  ParentRE          CHAR (128)    NOT NULL,
  ParentVersion     CHAR (16)     NOT NULL,
  StepID            CHAR (128)    NOT NULL,
  ParameterID       CHAR (32)     NOT NULL,
  ParentParamID     CHAR (32),
  ParameterValue    CHAR (128),
  DataInterpretation  INTEGER,
  Scaled            INTEGER,
  PRIMARY KEY (ParentRE, ParentVersion, StepID, ParameterID),
  FOREIGN KEY (ParentRE, ParentVersion, StepID)
    REFERENCES BXT_MRecipeStep (ParentRE, ParentVersion, StepID))

```

```

CREATE TABLE BXT_MRecipeOtherInformation (
  RE_ID             CHAR (128)    NOT NULL,
  REVersion         CHAR (16)     NOT NULL,
  StepID            CHAR (128)    NOT NULL,
  DataID            CHAR (32)     NOT NULL,
  DataType          CHAR (32),
  DataValue         CHAR (255),
  Description       CHAR (255),
  PRIMARY KEY (RE_ID, REVersion, DataID),
  FOREIGN KEY (RE_ID, REVersion)
    REFERENCES BXT_MRecipeElement (RE_ID, REVersion))

```

```

CREATE TABLE BXT_MRecipeElementEquip (
  RE_ID             CHAR (128)    NOT NULL,
  REVersion         CHAR (16)     NOT NULL,
  PropertyID        CHAR (32)     NOT NULL,
  DefaultValue      CHAR (128),
  DataInterpretation  INTEGER,
  EvaluationRule     INTEGER,
  EngrUnits         CHAR (32),
  Description       CHAR (255),
  PRIMARY KEY (RE_ID, REVersion, PropertyID),
  FOREIGN KEY (RE_ID, REVersion)
    REFERENCES BXT_MRecipeElement )

```

```

CREATE TABLE BXT_MRecipeStepEquip (
  ParentRE          CHAR (128)    NOT NULL,
  ParentVersion     CHAR (16)     NOT NULL,
  StepID            CHAR (128)    NOT NULL,
  PropertyID        CHAR (32)     NOT NULL,
  PropertyValue      CHAR (128),
  PRIMARY KEY (ParentRE, ParentVersion, StepID, PropertyID),
  FOREIGN KEY (ParentRE, ParentVersion, StepID)
    REFERENCES BXT_MRecipeStep (ParentRE, ParentVersion, StepID))

```

```
CREATE TABLE BXT_EquipElement (  
  EquipmentID      CHAR (32)      NOT NULL,  
  EE_Type          INTEGER,  
  EE_Level         INTEGER,  
  ContainedIn     CHAR (32),  
  Description      CHAR (255),  
  PRIMARY KEY (EquipmentID))
```

```
CREATE TABLE BXT_EquipLink (  
  EquipmentID      CHAR (32)      NOT NULL,  
  ToEquipmentID   CHAR (32)      NOT NULL,  
  Description      CHAR (255),  
  PRIMARY KEY (EquipmentID , ToEquipmentID),  
  FOREIGN KEY (EquipmentID)  
    REFERENCES BXT_EquipElement,  
  FOREIGN KEY (ToEquipmentID)  
    REFERENCES BXT_EquipElement )
```

```
CREATE TABLE BXT_EquipInclude (  
  EquipmentID      CHAR (32)      NOT NULL,  
  ClassEquipmentID CHAR (32)      NOT NULL,  
  Description      CHAR (255),  
  PRIMARY KEY (EquipmentID, ClassEquipmentID),  
  FOREIGN KEY (EquipmentID)  
    REFERENCES BXT_EquipElement,  
  FOREIGN KEY (ClassEquipmentID)  
    REFERENCES BXT_EquipElement )
```

```
CREATE TABLE BXT_EquipProperty (  
  EquipmentID      CHAR (32)      NOT NULL,  
  PropertyID       CHAR (32)      NOT NULL,  
  PropertyValue     CHAR (255),  
  EngrUnits        CHAR (32),  
  Description      CHAR (255),  
  PRIMARY KEY (EquipmentID, PropertyID),  
  FOREIGN KEY (EquipmentID)  
    REFERENCES BXT_EquipElement )
```

```
CREATE TABLE BXT_EquipInterface (  
  EquipmentID      CHAR (32)      NOT NULL,  
  EPI_ID           CHAR (32)      NOT NULL,  
  EPI_Definition   CHAR (32)      NOT NULL,  
  Description      CHAR (255),  
  PRIMARY KEY (EPI_ID, EquipmentID),  
  FOREIGN KEY (EquipmentID)  
    REFERENCES BXT_EquipElement )
```

```
CREATE TABLE BXT_EquipInterfaceDefinition (  
  EPI_Definition   CHAR (32)      NOT NULL,  
  Description      CHAR (255),  
  PRIMARY KEY (EPI_Definition) )
```

```

CREATE TABLE BXT_EquipInterfaceParameter (
  EPI_Definition      CHAR (32)      NOT NULL,
  ParameterID        CHAR (32)      NOT NULL,
  ParentParamID      CHAR (32),
  Type               INTEGER        NOT NULL,
  EngrUnits          CHAR (32),
  EnumSet            CHAR (32),
  Scaled             INTEGER,
  DefaultValue       CHAR (128),
  Description         CHAR (255),
  PRIMARY KEY (EPI_Definition, ParameterID),
  FOREIGN KEY (EPI_Definition)
    REFERENCES BXT_EquipInterfaceDefinition )

CREATE TABLE BXT_ScheduleEntry (
  ScheduleEntryID    CHAR (64)      NOT NULL,
  ParentSchedID      CHAR (64),
  ExternalID         CHAR (64),
  RE_ID              CHAR (128),
  REVersion          CHAR (16),
  SE_Type            INTEGER,
  BatchID            CHAR (128),
  LotID              CHAR (128),
  CampaignID         CHAR (128),
  ProductID          CHAR (32),
  OrderID            CHAR (128),
  SE_Action          INTEGER,
  SchedStatus        INTEGER,
  StartCondition     CHAR (255),
  InitialMode        INTEGER,
  SchedStartTime     DATETIME,
  SchedEndTime       DATETIME,
  BatchPriority       INTEGER,
  BatchSize          FLOAT,
  EngrUnits          CHAR (32),
  SENote             CHAR (255),
  Description         CHAR (255),
  PRIMARY KEY (ScheduleEntryID))

CREATE TABLE BXT_ScheduleEquip (
  ScheduleEntryID    CHAR (64)      NOT NULL,
  RequirementID      CHAR (32)      NOT NULL,
  Description         CHAR (255),
  PRIMARY KEY (ScheduleEntryID, RequirementID))

CREATE TABLE BXT_ScheduleProperty (
  ScheduleEntryID    CHAR (64)      NOT NULL,
  RequirementID      CHAR (32)      NOT NULL,
  PropertyName       CHAR (32)      NOT NULL,
  PropertyValue       CHAR (255),
  EngrUnits          CHAR (32),
  Description         CHAR (255),
  PRIMARY KEY (ScheduleEntryID, RequirementID, PropertyName))

```

```
CREATE TABLE BXT_ScheduleParameter (  
  ScheduleEntryID      CHAR (64)  NOT NULL,  
  ParameterID          CHAR (32)  NOT NULL,  
  ParentParameterID   CHAR (32),  
  ParameterValue       CHAR (255),  
  EngrUnits            CHAR (32),  
  ItemLocation         CHAR (128),  
  EnumSet              CHAR (32),  
  Description          CHAR (255),  
  PRIMARY KEY (ScheduleEntryID, ParameterID))
```

```
CREATE TABLE BXT_HistoryElement (  
  HistoryElementID     INTEGER  NOT NULL,  
  BatchID              CHAR (128),  
  MasterRecipeID      CHAR (128),  
  MasterRecipeVersion  CHAR (16),  
  ControlRecipeID     CHAR (28),  
  ReferenceEquipProcedure INTEGER,  
  RecipeProcedure     CHAR (128),  
  UnitProcedure       CHAR (128),  
  UnitProcedureCounter INTEGER,  
  Operation            CHAR (128),  
  OperationCounter    INTEGER,  
  Phase                CHAR (128),  
  PhaseCounter        INTEGER,  
  EquipmentID         CHAR (32),  
  EPI_ID              CHAR (32),  
  PRIMARY KEY (HistoryElementID))
```

```
CREATE TABLE BXT_HistoryLog (  
  RecordID             INTEGER  NOT NULL,  
  UTC                  DATETIME,  
  LocalTime           DATETIME  NOT NULL,  
  BatchID              CHAR (128),  
  HistoryElementID    INTEGER,  
  EquipmentID         CHAR (32),  
  EPI_ID              CHAR (32),  
  UserID              CHAR (64),  
  RecordSet           INTEGER  NOT NULL,  
  RecordSubSet        INTEGER,  
  RecordAlias         CHAR (32),  
  NewValue            CHAR (128),  
  OldValue            CHAR (128),  
  EngrUnits           CHAR (32),  
  PRIMARY KEY (RecordID))
```

Annex C (informative) — Abbreviations

The following are abbreviations that are used in this standard:

BXT	Batch exchange table
EPE	Equipment procedural element
ID	Identification
IEC	International Electrotechnical Commission
ISA	ISA–The Instrumentation, Systems, and Automation Society
ISO	International Organization for Standardization
MR	Master recipe
PFC	Procedure function chart
RE	Recipe element
SFC	Sequential function chart
SOP	Standard operating procedure
SQL	Structured query language
UML	Unified modeling language
UTC	Universal coordinated time

This page intentionally left blank.

Annex D (informative) — Language guidelines

A language is a set of symbols and the rules for their use in communication. Clause 6 describes guidelines for creating the symbols and rules that are needed for selected batch process-related communications.

Communication, involving machines and people, occurs among all six of the control activities that are described in Part 1. Both ends of any communication path must understand the same symbols and they must use the same rules.

Communication between people and electronic batch processing systems is usually done with a video or other graphic devices and various pointing and input devices. Text is frequently used for detailed communication, but visual symbols are often more effective for communicating complex relationships.

The central feature of batch control is the recipe. Graphic symbols related to the procedure and rules for their use are defined in clause 6. Other parts of the recipe (e.g., formula) contain detail that may be better represented visually as text.

D.1 PFC derivation

Three example depiction methods, Table format, Gantt Chart notation, and Sequential Function Charts (SFC), were discussed in ISA-TR88.00.03-1996, *Possible Recipe Procedure Presentation Formats*.

A table format is attractive because of its simplicity, intuitive interpretation, and flexibility (e.g., support of additional attributes listed in a tabular form, of inserts). However, the table format is limited to procedures that are linear in nature because it does not adequately address selections and parallel steps.

Gantt Chart notation is commonly used to depict time-oriented activities and it can be extended to depict recipe procedures in more complex cases than the table format. However, Gantt Chart notation does not lend itself to portraying conditional decisions.

Sequential function charts may be used to portray the conditional decisions in a procedure. However, there are certain aspects of procedures that are not adequately represented by SFCs or that become unnecessarily complicated to depict in SFCs.

Elements of these three depiction methods were combined to create a notation called Procedure Function Chart (PFC). The PFC is defined to graphically depict the procedure portion of the recipe and it is a derivation of Function Chart notation, as defined in IEC 60848:1988, that has been modified to make it usable in recipe depiction, and to add some of the benefits of Gantt Chart notation and the table format.

D.2 Recipe procedure

The separation of the recipe procedure (that defines desired process functionality) from the equipment procedure (that defines control execution) supports one of the goals stated clearly in Part 1: that recipes can be created without routine control engineering involvement. This goal defines a division of effort between the control engineering function and the recipe authoring function. This goal requires that the control engineering function defines and implements equipment procedural elements (e.g., equipment phases) and provides representations for use by the Master Recipe author, with appropriate constraints.

Part 1 defined the following four levels of procedural elements:

- a) Procedure
- b) Unit procedure

- c) Operation
- d) Phase

The procedure consists of some number of unit procedures. A unit procedure consists of an ordered set of operations that depict a contiguous production sequence that is to take place within a unit. An operation is made up of an ordered set of phases. Although there is no limit to the number of phases that may be simultaneously active in a unit, only one operation is presumed to be active in a unit at any time. Unit procedures are largely independent, but they encapsulate or reference lower-level operations and phases that may interact with operations and phases in other unit procedures.

D.3 Requirements for procedural control element depiction

A clear method of depicting procedural control elements in recipe procedures is essential. The following requirements have been identified for a clear depiction method:

- a) Simple to follow: Easy for people to understand
- b) Easy to build: Few syntax requirements and symbols to learn
- c) Clearly defined boundaries: Standardized graphical symbol for Start and End
- d) Unambiguous depiction of execution order: Sequence, parallelism, selection (divergence) and convergence
- e) Expression of coordination relationships: Material transfers, Wait for, Synchronize
- f) Hierarchical level: Standardized symbols for Procedure, Unit Procedure, Operation, Phase
- g) Existence of levels: Standardized graphical symbol to show possible decomposition of an element of the hierarchy
- h) Applicable to master recipes and control recipes
- i) Applicable to all levels: Similar set of symbols and rules at all levels in a recipe
- j) Independent of media: Equally usable and understandable whether implemented with pencil and paper or with full-color animated computer graphics

Annex E (informative) — Procedure function chart processing examples

Many rules concerning Procedure Function Chart processing are system dependent. The following examples illustrate possible PFC processing systems.

In these examples, a dot is used to identify the active symbols. The dot is not part of the symbols and it is only used for explanatory purposes. Active means that the symbols are being evaluated or executed by the PFC processing system.

When a procedural element is active, the lower-level PFC or equipment procedural element that it represents is currently being processed according to its state model. For these examples, the ANSI/ISA-88.01-1995 example state model is used. Inactive means that the symbols are not being evaluated or executed by the PFC processing system. When a procedural element is inactive, the lower-level PFC or equipment procedural element that it represents has a state that is appropriate with its state model, but it is not being processed and, therefore, it cannot change its state.

When an explicit transition is active, its expression is being evaluated by the PFC processing system, or the expression has already been evaluated true and notification of this has been sent to the lower-level PFC or equipment procedural entity that the preceding recipe element represents.

Example 1: Phase completes after the trailing transition is evaluated to be true (see following figure).

	<p>The PFC has started and the begin symbol is active.</p>
	<p>Recipe phase S1 becomes active. The corresponding equipment phase (not shown) has also become active.</p> <p>Explicit transition T1 becomes active and it is continually evaluated once recipe phase S1 becomes active.</p>
	<p>T1 is evaluated to be true.</p> <p>Equipment phase S1 receives input that T1 is true.</p> <p>Equipment phase S1 continues to be active until its logic results in it becoming inactive.</p>
	<p>Equipment phase S1 enters the complete state and it becomes inactive. This completion may be a result of the input from T1 and the successful accomplishment of housekeeping actions.</p> <p>Recipe phase S1 becomes inactive once equipment phase S1 becomes inactive.</p> <p>T1 stops being evaluated and it becomes inactive.</p> <p>Recipe phase S2 and its corresponding equipment phase S2 become active.</p> <p>Explicit transition T2 becomes active and it is continually evaluated.</p>

This page intentionally left blank.

Developing and promulgating technically sound consensus standards, recommended practices, and technical reports is one of ISA's primary goals. To achieve this goal, the Standards and Practices Department relies on the technical expertise and efforts of volunteer committee members, chairmen, and reviewers.

ISA is an American National Standards Institute (ANSI) accredited organization. ISA administers United States Technical Advisory Groups (USTAGs) and provides secretariat support for International Electrotechnical Commission (IEC) and International Organization for Standardization (ISO) committees that develop process measurement and control standards. To obtain additional information on the Society's standards program, please write:

ISA
Attn: Standards Department
67 Alexander Drive
P.O. Box 12277
Research Triangle Park, NC 27709