PSLX Engineering Specification

XML Specification

PSLX-04

<< DRAFT >>

Version 0.3

Update History

Date	Author	Version	Explanation
2003.7.23		0.1	Chap 2, 3 are translated to English
2003.7.28		0.2	Chap 5 is translated
2003.8.4		0.3	Chap 4 is translated.

Contents

＜Note＞

PSLX Consortium Japan and the members don't take on the responsibility for any losses caused from using this specification and the contents of this specification.

1. Intrduction

1.1. Purpose of This Specification

The purpose of this specification is to decide the rules for exchanging data between the business application program as an APS agent and the agent itself or other agents. When exchanging the data between some applications, it is necessary to decide the procedure for exchanging the data and how to describe the data to be exchanged.

This specification has the condition that data is exchanged based on XML and decides the procedure required for exchanging XML data and the way of describing XML tags. This specification describes the rules that must be strictly kept or are recommended to keep.

Each APS agent can be linked only when the software of APS agent follows the rules and then the collaboration that APS aims at is realized.

1.2. Intended Reader

The intended readers of this specification are as below.

Engineering staff at IT section of manufacturing enterprise, Engineering staff of SI enterprise, Manager of software package vendor, Engineering staff of software package vendor, Student in manufacturing management

1.3. Structure of Specification

The structures of the next and later chapters are as below.

Part 1：Structure of XML Tag (Chapter 2) and How to Describe XML Tag (Chapter 3).

Part 2：PSLX Messaging Specification (Chapter 4).

Part 3：XML Data Exchange Specification (Chapter 5).

1.4. Notes for Using This Specification

✧ **This is a snap shot of business basically.**

The design does not consider various situations of usage changing in the time series, not like a database. The models that can be expressed by the tag constructure defined in this specification are just snap shots of business. In short, the constructure aims at transmitting information from one business application to itself or another business application.

✧ **Measures against multivocal data according to process**

Even if the elements and structure of data are the same in every business process in manufacturing enterprise, the meaning of data depends on the situation where the message is sent or received. In short, the meaning of data depends on the situation where the data is used. Therefore it is necessary to use a tag according to the type of interface as a rule, not only the grammer of tag.

✧ **Converting ontology to business terms**

The tag terms used in this specification are the ontology for exchanging data between different businesses. The terms that are usually used in individual business or enterprise may be different from the tag terms in this specification. In such a case, associate each term with a tag name in this specification and then use it.

✧ **Dividing specification according to layer**

The rule at transport layer level in communication, the rule at communication software level using it, and the rule at application level are required for two different applications to exchange data. This specification mainly aims at the rule at application level. Refer to de facto standard or de juri standard for other levels.

1.5. Indexes for Phased Implementation

The concept, implementation level is provided for actually implementing the content of this specification in a business application program as a PSLX interface. Each application can

communicate with each other according to the level by declaring each implementation level.

The software implementing a PSLX interface must declare the following contents.

✧ **Communication protocol**

The software shows which of communication protocols such as HTTP or SMTP is selected. The contents are described when the protocol follows the communication rule at XML level decided under the above communication protocols such as SOAP and ebXML etc.

✧ **PSLX communication method**

The communication methods under the PSLX specification are synchronous communications and asynchrous communications. Asynchrouns communications are Push-Push type, Push-Pull type, and Pull-Push type. The applicable communication method is declared. In case of asynchronous communication, it is possible to declare whether the request to receive can be set up or not. When not declaring, it is regareded that the request to receive can be set up.

✧ **Name of interface to be implemented**

It is not necessary to implement all the interfaces defined in this specification. Each application can select the interface to be implemented out of them. However when the interface not to be implemented is called, the error message notifying that the interface is not implemented must be returned.

✧ **Level of implementation interface**

There are levels of implementing from level 1 to level 3 for each interface. Level 1 is a required minimum function. Level 3 indicates the full function. This specification shows the concrete contents to be achieved on each level.

1.6. Policy on Copying Specification

It is free to copy this specification and distribute the copies. But it is

prohibited to modify the contents of this specification. When referring to a part or all the parts of the contents of this specification for another document, write URL（http://www.pslx.org/）of the applicable item on the WWW site of PSLX Consortium Japan preserving this specification.

2. Structure of XML Tag（Part１）

This chapter explains the specifications of all the tags used in PSLX. The contents given in this chapter are in http://www.pslx.org/schema/core/1.0/ as XML schema with the below names.

Implementation level	File name
1	PSLXSchema1.xsd
2	PSLXSchema2.xsd
3	PSLXSchema.xsd

2.1 PSLX Name Space

The tag sets of PSLX can be distinguished by a name space. URI showing the name spaces of PSLX is as below.

```
http://www.pslx.org/schema/core/1.0
```

PSLX is described by using a name space as below.

```
<pslx:pslx xmlns:pslx="http://www.pslx.org/schema/core/1.0">
   <pslx:profile name="production control problem 1"/>
   <pslx:order name="K01"/>
</pslx>
```

2.2. Definition of Control Data

2.2.1. The Highest Class Data（pslx）

This is the tag in the highest class of PSLX data. This tag is used as the top level for filing PSLX data and for sending and receiving PSLX data between applications.

Tag name	pslx		Implementation level		1
Upper tag	No				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual

type	string	request | response | receipt | exception | file		file	1
id	string				1
ref	string				1
action	string				1
receipt	boolean			false	2
Element name	Explanation		min	max	Actual
error	Error or warning information		0	No	1
spec	Specification information on service		0	No	1
unit	Definition information on unit		0	No	3
scale	Scale information on time		0	No	2
shift	Capacity pattern information		0	No	3
profile	Information on the whole PSLX data		0	1	1
customer	Information on customer		0	No	1
supplier	Information on supplier		0	No	1
item	Information on item		0	No	1
resource	Information on resource		0	No	1
lot	Information on lot		0	No	1
task	Information on task		0	No	1
event	Information on event		0	No	1
operation	Information on operation		0	No	1
order	Information on order		0	No	1
parameter	Information on subject		0	No	1
expression	Information on constraint		0	No	1
Illegal code	Illegal contents				Actual

	If type attribute is response, exception or receipt, ref attribute must be set up.	
	Id must be unique in every application making a message.	
	If ref is set up, there must be the applicable message.	
	The interface name prescribed beforehand must be set up in action.	
	If type attribute is receipt or exception, receipt attribute must be false.	

2.2.2. Profile Data（profile）

This is the tag to set up the information about the whole PSLX data. Especially when filing PSLX data, the author and the creation date as attribute information of the stored data are set up in this data.

Tag name	profile		Implementation level		1
Upper tag	pslx(1)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
name	string				1
author	string				1
version	string				1
create	datetime				1
expire	datetime				1
current	datetime				1
start	datetime				1
end	datetime				1
Element name	Explanation		min	max	Actual
description	Note information		0	1	1
display	How to display schedule		0	1	2
priority	Priority of schedule		0	1	2
spec	Optional specification		0	No	2
Illegal code	Illegal contents				Actual

	The end date must be always later than the start date. (The same time is prohibited.)	
	The current date must be between start and end（including start and end）.	

2.2.3. Error Data（error）

This is the data to notify the error content to the party when any fault like an error occurs. There are “warning” showing that the operation can be continued, and “error” showing that the operation cannot be continued.

Tag name	error		Implementation level		1
Upper tag	pslx(1)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
code	string				1
severity	string	warning | error		error	1
Element name	Explanation		min	max	Actual
No					
Illegal code	Illegal contents				Actual
No					

2.2.4. Display Color Data（color）

This data specifies a display color for displaying a basic element on a device. A numerical value of RGB is specified for the color.

Tag name	color		Implementation level		2
Upper tag	display(2)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
r	int	Integers from 0 to 255	○		2
g	int	Integers from 0 to 255	○		2

b	int	Integers from 0 to 255	○		2
Element name	Explanation		min	max	Actual
No					
Illegal code	Illegal contents				Actual
No					

2.2.5. Display Data（display）

This data specifies how to display a basic element on a screen or a document through GUI. The existence of display, a display location and a display color can be specified.

Tag name	display		Implementation level		2
Upper tag	profile(2), address(2), event(2), ev(2), start(2), end(2), release(2), duetime(2), customer(2), supplier(2), item(2), resource(2), operation(2), order(2), lot(2), task(2), parameter(3), expression(3),				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
show	boolean			true	2
name	string				2
row	long			0	3
col	long			0	3
scale	IDREF	Referring to scale			3
Element name	Explanation		min	max	Actual
color	Color of the displayed element in the upper class		0	1	2
Illegal code	Illegal contents				Actual
No					

2.3. Definition of Data Related with Unit

2.3.1. Unit Data（unit）

The various values and units of time information are set up. The data reserved beforehand are #sec, #min, #hour, #day, #week, #10d, #15d, #month, and #year for time.

Tag name	unit		Implementation level		3
Upper tag	pslx(3)				
Reference	scale(2), qty(1), translate(3), lotsize(3), tasksize(3), price(1), duration(3), task(3), order(3), expression(2), parameter(2),				
Attribute	Type	Constraint	Required	Omitted	Actual
name	ID	Reference key	○		3
Element name	Explanation		min	max	Actual
translate	How to translate data between unit systems		0	No	3
Illegal code	Illegal contents				Actual
No					

2.3.2. Unit Translation Data（translate）

The way of translating unit between some unit systems is set up. This tag is defined in the element of translation origin. The element tags are listed in every unit of translation destination in which the unit can be translated. The specified unit can be translated into the unit specified with an upper tag.

Tag name	translate		Implementation level		3
Upper tag	unit(3)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
unit	IDREF	Referring to unit key	○		3
value	double		○		3
origin	double			0	3

Element name	Explanation	min	max	Actual
base	Denominator of a fractional conversion factor	0	1	3
Illegal code	Illegal contents			Actual
No				

2.3.3. Time Conversion Data（scale）

The way of converting date data into a real number is specified. This data is used for converting schedule into a location on the screen or changing an absolute date into a relative value.

Tag name	scale		Implementation level		2
Upper tag	pslx(2)				
Reference	time(3), display(3)				
Attribute	Type	Constraint	Required	Omitted	Actual
name	ID	Reference key	○		2
unit	IDREF	Referring to unit element			2
value	double	value > 0		1	2
origin	datetime				2
type	string	disc | cont		cont	3
Element name	Explanation		min	max	Actual
No					
Illegal code	Illegal contents				Actual
	The value of unit must correspond to a time unit such as sec | min | hour | day | week | 10d | 15d | month | year and so on.				

2.4. Definition of Numerical Data

2.4.1. Numerical Data（qty）

The various numerical data used for planning or scheduling are set up. The additional information such as unit can be set up.

Tag name	qty		Implementation level		1
Upper tag	spec(1), progress(1), capacity(2), load(1), stock(1), produce(1), consume(1), assign(1), partof(2), pegging(2), tracking(2), condition(2), action(3), lot(1), task(1), operation(2), order(1)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
value	double				1
unit	IDREF	Referring to unit element			1
Element name	Explanation		min	max	Actual
base	Denominator to express a numerical value by a fraction		0	1	3
min	Minimum value to set limits		0	1	2
max	Maximum value to set limits		0	1	2
Illegal code	Illegal contents				Actual
No					

2.4.2. Price Data（price）

This data expresses the information that can be converted into a sum of money such as price or cost. The kind of currency can be set up as a unit.

Tag name	price	Implementation level	1

Upper tag	item(1), resource(2), lot(2), task(2), order(1), operation(2), event(2), ev(2), start(2), end(2), release(2), duetime(2),				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
value	double				1
unit	IDREF	Referring to unit element			1
Element name	Explanation		min	max	Actual
base	Denominator to express a price by a fraction		0	1	3
min	Minimum value to set limits		0	1	2
max	Maximum value to set limits		0	1	2
Illegal code	Illegal contents				Actual
No					

2.4.3. Denominator of Numerical Value（base）

This data is used for expressing a numerical value by a fraction. The data is used when a real number reduces the precision of value.

Tag name	base		Implementation level		3
Upper tag	qty(3), price(3), translate(3)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
value	double				3
Element name	Explanation		min	max	Actual
No					
Illegal code	Illegal contents				Actual
No					

2.4.4. Priority Data（priority）

Priority data is set up. The real numbers from –1 to 1 are used as values. The nearer to 1 the number is, the higher the priority is.

Tag name	priority		Implementation level		1
Upper tag	profile(2), address(2), event(2), ev(2), start(2), end(2), release(2), duetime(2), customer(1), supplier(2), item(2), resource(2), lot(1), task(2), order(1), operation(1), produce(2), consume(2), assign(2), lotsize(3), tasksize(3), changeover(2), interval(3), parameter(2), expression(2)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
value	double	-1≦x≦1			1
Element name	Explanation		min	max	Actual
min	Minimum value of priority		0	1	2
max	Maximum value of priority		0	1	2
Illegal code	Illegal contents				Actual
No					

2.5. Definition of Character Data

2.5.1. Character Data（char）

This data is used for expressing the information by characters or symbol data as a value, not a numerical value. The value is an optional character string.

Tag name	char		Implementation level		1
Upper tag	spec(1), condition(2), action(3)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
value	string				1
Element name	Explanation		min	max	Actual

enumerate	Proposed element in case that there are multiple proposed elements.	0	No	3
Illegal code	Illegal contents			Actual
No				

2.5.2. Address Data (address)

Address data is set up. The value of address data can be specified with a character string or URI. The value can be specified by a coordinate system such as east, south and height to express a location by a numerical value.

Tag name	address		Implementation level		1
Upper tag	customer(1), supplier(1), location(3), produce(3), consume(3), assign(3), condition(3), action(3), lot(3), task(3),				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
value	string				1
uri	URI				1
east	double				2
south	double				2
height	double				2
Element name	Explanation		min	max	Actual
description	Note to location		0	1	1
display	How to display location		0	1	2
priority	Priority of location		0	1	2
spec	Specification added to the location		0	No	1
Illegal code	Illegal contents				Actual
No					

2.5.3. Note Data（description）

This data is used for adding notes to various elements. The content of note is an optional character string.

Tag name	description		Implementation level		1
Upper tag	profile(1), address(1), event(1), ev(2), start(2), end(2), release(2), duetime(2), customer(1), supplier(1), item(1), resource(1), operation(1), order(1), lot(1), task(1), parameter(1), expression(1),				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
No					
Type	Contents				
string	Contents of note				
Illegal code	Illegal contents				Actual
No					

2.6. Definition of Time Data

2.6.1. Date Data（time）

This is the data to set up the date and time. The time when an event occurs and the time information of time series information are specified with a date. The discrete date and time (term) can also be expressed.

Tag name	time		Implementation level		1
Upper tag	progress(1), capacity(2), load(1), stock(1), spec(2), location(3), event(1), ev(2), start(1), end(1), release(1), duetime(1), lot(2), task(2),				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
ref	string	#init | #final | #now			1
value	datetime				1
count	long				3

scale	IDREF	Referring to scale key			3
Element name	Explanation		min	max	Actual
earliest	Earliest date and time		0	1	2
latest	Latest date and time		0	1	2
Illegal code	Illegal contents				Actual
	Only one of ref, value and count can be specified.				

2.6.2. Time Data（duration）

Time data is set up. The date indicates the time point of one location, while the time data indicates the relation between two dates.

Tag name	duration		Implementation level		1
Upper tag	predecessor(1), successor(2), changeover(2), interval(3), operation(1),				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
value	duration				1
count	long				3
unit	IDREF	Referring to unit key			3
type	string	fixed | longer | shorter		longer	2
net	boolean			false	3
Element name	Explanation		min	max	Actual
shortest	Shortest time		0	1	2
longest	Longest time		0	1	2
Illegal code	Illegal contents				Actual

2.7. Definition of Time Series Data

2.7.1. Specification Data（spec）

This data is used for setting up optional specifications for various basic elements. The type of specification can be originally set up as a specification name. Both a character string and a numerical value can be used as a value of specification. This data can indicate the change of time in specification.

Tag name	spec		Implementation level		1
Upper tag	pslx(1), profile(2), address(1), event(1), ev(2), start(2), end(2), release(2), duetime(2), customer(1), supplier(1), item(1), resource(1), order(1), operation(1), lot(1), task(1), parameter(2), expression(2),				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
name	string				1
type	string	disc | cont		disc	2
relative	boolean			false	2
Element name	Explanation		min	max	Actual
time	Date and time when the option data becomes the specified value		0	1	2
char	Value of option data（character string）		0	1	1
qty	Value of option data（numerical value）				1
Illegal code	Illegal contents				Actual
	The numerical option data and the character option data may not be mingled in the combination of element type and option name.				
	The character string starting with # cannot be set up in an option name.				

	When relative is false, this data must be unique in the name value and the time value under the same element.	
	The options with the same name under the same element must be sorted in order of early date of time.	

2.7.2. Location Data（location）

Location data is set up. Address data indicates individual location data, while this element indicates the situation where address changes in time series. Thus location data has a pair of address data and its date data.

Tag name	location		Implementation level		3
Upper tag	item(3), resource(3),				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
type	string	disc | cont		disc	3
Element name	Explanation		min	max	Actual
time	Date and time when the location data is valid		0	1	3
address	Address information		0	1	3
Illegal code	Illegal contents				Actual
	When there are multiple location data in the upper element, the date must be sorted in order of early date.				

2.7.3. Progress Data（progress）

Progress of operation and order is specified. Progress data can be specified in every setting date and time (measuring date and time). As the time passes, the state that progress changes can be expressed.

Tag name	progress	Implementation level	1
Upper tag	operation(1), order(1)		
Reference			

Attribute	Type	Constraint	Required	Omitted	Actual
status	string	completed | canceled | suspended | started | ready | created			1
relative	boolean			false	2
Element name	Explanation		min	max	Actual
time	Date and time when progress is set up		0	1	1
qty	Progress information on the quantity of production and so on		0	1	1
Illegal code	Illegal contents				Actual
	When there are multiple progress data in the upper element, the data must be sorted in order of early date.				

2.7.4. Capacity Data（capacity）

This data shows the information about the production capacity changing in every date and time. The capacity data is set up with the date information. This data can express setting the upper and the lower limits on load to resource, and the constraint on the stock level to item.

Tag name	capacity		Implementation level		2
Upper tag	item(2), resource(2), shift(3)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
type	string	disc | cont		disc	2
direction	string	upper | lower		upper	2
relative	boolean			false	2
Element name	Explanation		min	max	Actual

time	Date and time when the capacity is set up	0	1	2
qty	Value of the capacity to be set up	0	1	2
Illegal code	Illegal contents			Actual
	When there are multiple capacity data in the upper element, the data must be sorted in order of early date.			

2.7.5. Load Data（load）

The state of load to resource is expressed in time series. Generally when the operation using resource starts, the level of load rises and after the operation ends, the level gets down to normal. Load data can express how many resources are used by some operations at each point of time.

Tag name	load		Implementation level		1
Upper tag	resource(1),				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
type	string	disc | cont		disc	2
relative	boolean			false	2
Element name	Explanation		min	max	Actual
time	Date when the load is set up		0	1	1
qty	Value of the load to be set up		0	1	1
Illegal code	Illegal contents				Actual
	When there are multiple load data in the upper element, the data must be sorted in order of early date.				

2.7.6. Stock Data（stock）

The quantity of item stocks is set up in time series. The level of stock is changed by the operation producing the item and the operation consuming the item. The amount of stocks at each point of time can be expressed by the stock data.

Tag name	stock		Implementation level		1
Upper tag	item(1),				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
type	string	disc | cont		disc	2
relative	boolean			false	2
Element name	Explanation		min	max	Actual
time	Date and time when the stock is set up		0	1	1
qty	Value of the stock to be set up		0	1	1
Illegal code	Illegal contents				Actual
	When there are multiple stock data in the upper element, the data must be sorted in order of early date.				

2.8. Definition of Calendar Data

2.8.1. Shift Pattern（shift）

This is the element to set the pattern information of capacity. Capacity tag as capacity data is defined in this element and the content is named as a pattern. This data is used for setting up a calendar.

Tag name	shift		Implementation level		3
Upper tag	pslx(3)				
Reference	calendar(3)				
Attribute	Type	Constraint	Required	Omitted	Actual
name	ID	Reference key	○		3

Element name	Explanation	min	max	Actual
capacity		1	No	3
Illegal code	Illegal contents			Actual
	Capacity element must be sorted in order of early date.			

2.8.2. Calendar Data（calendar）

This is the data to use a shift pattern when defining capacity information related with resource or item. The information specified as calendar data can be changed into the capacity data based on the shift pattern information.

Tag name	calendar		Implementation level		3
Upper tag	item(3), resource(3),				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
shift	IDREF	Referring to shift key	○		3
time	datetime		○		3
Element name	Explanation		min	max	Actual
No					
Illegal code	Illegal contents				Actual
	When there are multiple calendar data in the upper element, the data must be sorted in order of early date of time element.				

2.9. Definition of Relation Attribute Data

2.9.1. Production Data（produce）

The item produced by operation is specified. Some production items can be specified for one operation. And a production item may not be set up.

Tag name	produce		Implementation level		1
Upper tag	operation(1), event(1), ev(3), start(3), end(3), release(3), duetime(3)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
type	string	disc | cont		disc	2
nth	int			1	2
sel	int			1	3
item	IDREF	Referring to item key			1
lot	IDREF	Referring to lot key			1
Element name	Explanation		min	max	Actual
priority	Priority		0	1	2
qty	Quantity of production		0	1	1
address	Production address		0	1	3
Illegal code	Illegal contents				Actual
	The value of value attribute of qty must be plus.				
	Either item attribute or lot attribute must be set up.				
	The upper tag must be instance information for specifying lot attribute.				
	nth and sel must be serial integers from 1 in the valid range.（nth is within this element and sel is within the upper element.）				

2.9.2. Consumption Data（consume）

When item is consumed by operation or event, the content is set up. This corresponds to the case where a raw material for operation is consumed by operating. Some consumption data can be specified for the operation or the event to be set up.

Tag name	consume	Implementation level	1

Upper tag	operation(1), event(1), ev(3), start(3), end(3), release(3), duetime(3)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
type	string	disc | cont		disc	2
nth	int			1	2
sel	int			1	3
item	IDREF	Referring to item key			1
lot	IDREF	Referring to lot key			1
Element name	Explanation		min	max	Actual
priority	Priority		0	1	2
qty	Quantity of consumption		0	1	1
address	Consumption address		0	1	3
Illegal code	Illegal contents				Actual
	The value of value attribute of qty must be plus.				
	Either item attribute or lot attribute must be set up.				
	The upper tag must be instance information for specifying lot attribute.				
	nth and sel must be serial integers from 1 within the valid range.（nth is within this element and sel is within the upper element.）				

2.9.3. Assignment Relation（assign）

The situation where operation uses resource is set up. The amount of used resources can be specified at the same time. The constraint of alternative resources that shows which resource can be used by operation can be also expressed.

Tag name	assign	Implementation level	1
Upper tag	operation(1),		
Reference			

Attribute	Type	Constraint	Required	Omitted	Actual
type	string	disc | cont		disc	2
nth	int			1	2
sel	int			1	3
resource	IDREF	Referring to resource key			1
task	IDREF	Referring to task key			1
Element name	Explanation		min	max	Actual
priority	Priority		0	1	2
qty	Quantity of assignment		0	1	1
address	Assignment address		0	1	3
Illegal code	Illegal contents				Actual
	The upper tag must be instance information for specifying task attribute.				
	nth and sel must be serial integers from 1 within the valid range.（nth is within this element and sel is within the upper element.）				

2.9.4. Predecessor Data（predecessor）

The interval between two operations is set up. The specified operation is a predecessor operation. Predecessor is the constraint that one operation can be started only after the other operation is ended. Predecessor can be set up between events or between operation and event.

Tag name	predecessor		Implementation level		1
Upper tag	operation(1), event(1), ev(3), start(3), end(3), release(3), duetime(3)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
type	string	FS | SS | FF | SF		FS	2
operation	IDREF	Referring to			1

		operation			
event	IDREF	Referring to event			3
Element name	Explanation		min	max	Actual
duration	Interval between predecessor and this element		0	1	1
Illegal code	Illegal contents				Actual
	Either operation or event must be set up.				
	The reference destination of event should be an ID of event element or an identification name of ev, start, end, release, or duetime.				

2.9.5. Successor Data（successor）

The interval between two operations is specified. The specified element is a successor operation or a successor element to the upper operation or event. This data is used for defining predecessor from the side of predecessor. Both predecessor and successor indicate the complete same contents.

Tag name	successor		Implementation level		2
Upper tag	operation(2), event(2), ev(3), start(3), end(3), release(3), duetime(3)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
type	string	FS | SS | FF | SF		FS	2
operation	IDREF	Referring to operation			2
event	IDREF	Referring to event			3
Element name	Explanation		min	max	Actual
duration	Interval between predecessor and this element		0	1	2

Illegal code	Illegal contents	Actual
	Either operation or event should be set up.	
	The reference destination of event should be an ID of event element or an identification name of ev, start, end, release, or duetime.	

2.10. Relation Information between Elements

2.10.1. Inclusion Data（partof）

Inclusion relation is set up for item, resource, operation and order. The whole element in inclusion relation is specified from the side of partial element. It doesn't matter that there is more than one element for the whole.

Tag name	partof		Implementation level		2
Upper tag	operation(2), item(2), resource(2), order(2)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
operation	IDREF	Reference key of operation			2
item	IDREF	Reference key of item			2
resource	IDREF	Reference key of resource			2
order	IDREF	Reference key of order			2
Element name	Explanation		min	max	Actual
qty			0	1	2
Illegal code	Illegal contents				Actual
	The value must be set up in one of operation, item, resource, and order.				
	The element to be specified must be the same as the type of upper element.				

	Reference may not be a loop.	

2.10.2. Pegging Data（pegging）

Pegging relation between orders is expressed. This data is used for linking and associating the orders from the origin order such as a customer order to the lower order finally corresponding to schedule.

Tag name	pegging		Implementation level		2
Upper tag	order(2)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
order	IDREF	Reference key of order	○		2
type	string	produce | consume		produce	2
final	boolean			false	3
Element name	Explanation		min	max	Actual
qty	Quantity of pegging		0	1	2
Illegal code	Illegal contents				Actual
	Reference may not be a loop.				

2.10.3. Tracking Data（tracking）

This data indicates the correspondence between lots or between tasks. This data manages which production lot corresponds to which consumption lot. This correspondence is used for lot tracking. When executing one task with dividing the task into some resources, the correspondence is managed.

Tag name	tracking		Implementation level		2
Upper tag	lot(2), task(3),				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
lot	IDREF	Reference key of lot			2

task	IDREF	Reference key of task			3
Element name	Explanation		min	max	Actual
qty			0	1	2
Illegal code	Illegal contents				Actual
	The value must be set up in either lot or task.				
	The element to be set up must be the same as the type of upper element.				
	Reference may not be a loop.				

2.11. Definition of Extended Constraint Data

2.11.1. Lot Size Data（lotsize）

Constraint information on lot size is defined. The constraint specified in this data is considered for producing lots. When this element isn't specified, the number of required elements is produced as one lot.

Tag name	lotsize		Implementation level		3
Upper tag	item(3), operation(3), order(3)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
unit	IDREF	Referring to unit key			3
type	string	fixed | min | max | unit		fixed	3
value	double				3
Element name	Explanation		min	max	Actual
priority	Importance of lot size constraint		0	1	3
Illegal code	Illegal contents				Actual

No		

2.11.2. Task Size Data（tasksize）

Task size is set up like lot size. This data is set up when the resource has a capacity as a total value but a unit of operation that can be executed at the same time is required.

Tag name	tasksize		Implementation level		3
Upper tag	resource(3), operation(3), order(3)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
unit	IDREF	Referring to unit key			3
type	string	fixed | min | max | unit		fixed	3
value	double				3
Element name	Explanation		min	max	Actual
priority	Importance of task size constraint		0	1	3
Illegal code	Illegal contents				Actual
No					

2.11.3. Condition（condition）

Condition is defined. This data is used as an executable condition of event, resource switch information or applicable condition of item existence information. Specifications of item and resource are referred.

Tag name	condition		Implementation level		2
Upper tag	changeover(2), interval(3), event(3), ev(3), start(3), end(3), release(3), duetime(3)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
item	IDREF	Referring to item			2

resource	IDREF	Referring to resource			2
spec	string		○		2
type	string	EQ | NE | GT | GE | LT | LE		NE	2
direction	string	pre | suc | cmb		cmb	2
Element name	Explanation		min	max	Actual
char	Character string to be compared		0	1	2
qty	Numerical value to be compared				2
address	Address to be compared				3
Illegal code	Illegal contents				Actual
	The value of spec attribute must be a spec name of the specified item or resource, or #stock, #load, #capacity, or #address.				
	The data of the comparison origin and the data of the comparison destination must be in agreement on qty, charm or address.				

2.11.4. Action（action）

The condition after the event is executed is set up. When action is executed, the content defined in this data is reflected. The target for operation is the value of specification of item or resource.

Tag name	action		Implementation level		3
Upper tag	event(3), ev(3), start(3), end(3), release(3), duetime(3)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
item	IDREF	Referring to item			3
resource	IDREF	Referring to resource			3

spec	string		○		3
relative	boolean			false	3
Element name	Explanation		min	max	Actual
char	Character string to be compared		1	1	3
qty	Numerical value to be compared				3
address	Address to be compared				3
Illegal code	Illegal contents				Actual
	The value of spec attribute should be a spec name of the specified item or resource, or #stock, #load, #capacity, or #address.				
	When relative is true, east, south, height of <qty> or <address> must be set up.				

2.11.5. Switch Data（changeover）

The relation between two operations continuously executed on the resource is defined. When two operations fulfill the specific condition, switch time or switch operation for setup occurs.

Tag name	changeover		Implementation level		2
Upper tag	resource(2),				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
operation	IDREF	Referring to operation key			3
Element name	Explanation		min	max	Actual
priority	Priority of setup		0	1	2
condition	Condition for comparison		0	No	2
duration	Time required for switching setup		0	1	2

Illegal code	Illegal contents	Actual
No		

2.11.6. Existence Data（interval）

The constraint on the time from producing item to consuming item is defined. When the specific condition is fulfilled between two serial operations, the constraint of interval time can be set up by the maximum value or the minimum value.

Tag name	interval		Implementation level		3
Upper tag	item(3),				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
operation	IDREF	Referring to operation key			3
Element name	Explanation		min	max	Actual
priority	Priority of setup		0	1	3
condition	Condition for comparison		0	No	3
duration	Time required for switching setup		0	1	3
Illegal code	Illegal contents				Actual
No					

2.12. Definition of Event Data

2.12.1. Event Data（event）

The independent event is defined. Event is the occurrence changing the state and occurs at the point of time. Event has only one date and time. The events such as start and end of operation, order duetime, and order issue are separately defined as special examples of event.

Tag name	event	Implementation level	1
Upper tag	pslx(1), parameter(1),		

Reference	predecessor(3), successor(3), event(2),				
Attribute	Type	Constraint	Required	Omitted	Actual
name	ID	Reference key	○		1
ac	string	create | delete | revise			1
calculation	string	sum | ave | cnt | min | max | earliest | latest | shortest | longest			3
master	boolean			false	2
parent	IDREF	Referring to event key			2
resource	IDREF	Referring to resource key			1

Element name	Explanation	min	max	Actual
query	How to inquire event	0	No	1
description	Note on event	0	1	1
display	How to display event	0	1	2
priority	Priority of event	0	1	2
spec	Optional specification on event	0	No	1
price	Price of event	0	1	2
time	Occurrence date and time of event	0	1	1
predecessor	Predecessor event or operation	0	No	1
successor	Successor event or operation	0	No	2
produce	Item to be produced	0	No	1
consume	Item to be consumed	0	No	1
condition	Condition	0	No	3
action	Execution result	0	No	3

Illegal code	Illegal contents	Actual
	The reference destination of parent must be master.	

2.12.2. Event Data in Operation (ev)

The event added to the specific operation is defined. The event such as suspend or restart other than start and end of operation corresponds to this data.

Tag name	ev		Implementation level		2
Upper tag	operation(2),				
Reference	predecessor(3), successor(3)				
Attribute	Type	Constraint	Required	Omitted	Actual
name	ID	Reference key	○		2
Element name	Explanation		min	max	Actual
description	Note on event		0	1	2
display	How to display event		0	1	2
priority	Priority of event		0	1	2
spec	Optional specification on event		0	No	2
price	Price of event		0	1	2
time	Occurrence date and time of event		0	1	2
predecessor	Predecessor event or operation		0	No	3
successor	Successor event or operation		0	No	3
produce	Item to be produced		0	No	3
consume	Item to be consumed		0	No	3
condition	Condition		0	No	3
action	Execution result		0	No	3
Illegal code	Illegal contents				Actual
	Name should be unique in the upper operation.				

	"Upper operation name # event name" should be unique in the upper operation name and this event.	

2.12.3. Start Event Data（start）

The event equivalent to operation start is set up. Start is one event and the date and time are set up in the same form as other events.

Tag name	start		Implementation level		1
Upper tag	operation(1)				
Reference	predecessor(3), successor(3)				
Attribute	Type	Constraint	Required	Omitted	Actual
No					
Element name	Explanation		min	max	Actual
description	Note on event		0	1	2
display	How to display event		0	1	2
priority	Priority of event		0	1	2
spec	Optional specification on event		0	No	2
price	Price of event		0	1	2
time	Occurrence date and time of event		0	1	1
predecessor	Predecessor event or operation		0	No	3
successor	Successor event or operation		0	No	3
produce	Item to be produced		0	No	3
consume	Item to be consumed		0	No	3
condition	Condition		0	No	3
action	Execution result		0	No	3
Illegal code	Illegal contents				Actual
	"Upper operation name #start" should be unique in the upper operation name and this event.				

2.12.4. End Event Data（end）

The event equivalent to operation end is set up. End is one event and the date and time are set up in the same form as other events.

Tag name	end		Implementation level		1
Upper tag	operation(1)				
Reference	predecessor(3), successor(3)				
Attribute	Type	Constraint	Required	Omitted	Actual
No					
Element name	Explanation		min	max	Actual
description	Note on event		0	1	2
display	How to display event		0	1	2
priority	Priority of event		0	1	2
spec	Optional specification on event		0	No	2
price	Price of event		0	1	2
time	Occurrence date and time of event		0	1	1
predecessor	Predecessor event or operation		0	No	3
successor	Successor event or operation		0	No	3
produce	Item to be produced		0	No	3
consume	Item to be consumed		0	No	3
condition	Condition		0	No	3
action	Execution result		0	No	3
Illegal code	Illegal contents				Actual
	“Upper operation name #end” should be unique in the upper operation name and this event.				

2.12.5. Order Release Event（release）

The event that an order can start such as material arrival is set up. Order release event is one event and the date and time are set up in the same form as other events.

Tag name	release		Implementation level		1
Upper tag	order(1)				
Reference	predecessor(3), successor(3)				
Attribute	Type	Constraint	Required	Omitted	Actual
No					
Element name	Explanation		min	max	Actual
description	Note on event		0	1	2
display	How to display event		0	1	2
priority	Priority of event		0	1	2
spec	Optional specification on event		0	No	2
price	Price of event		0	1	2
time	Occurrence date and time of event		0	1	1
predecessor	Predecessor event or operation		0	No	3
successor	Successor event or operation		0	No	3
produce	Item to be produced		0	No	3
consume	Item to be consumed		0	No	3
condition	Condition		0	No	3
action	Execution result		0	No	3
Illegal code	Illegal contents				Actual
	“Upper order name #release” should be unique in the upper order name and this event.				

2.12.6. Order Duetime Event（duetime）

The event indicating the latest end date of all the operations related with order such as product shipping is set up. Order duetime event is one event and the date and time are set up in the same form as other events.

Tag name	duetime	Implementation level	1
Upper tag	order(1)		
Reference	predecessor(3), successor(3)		

Attribute	Type	Constraint	Required	Omitted	Actual
No					
Element name	Explanation		min	max	Actual
description	Note on event		0	1	2
display	How to display event		0	1	2
priority	Priority of event		0	1	2
spec	Optional specification on event		0	No	2
price	Price of event		0	1	2
time	Occurrence date and time of event		0	1	1
predecessor	Predecessor event or operation		0	No	3
successor	Successor event or operation		0	No	3
produce	Item to be produced		0	No	3
consume	Item to be consumed		0	No	3
condition	Condition		0	No	3
action	Execution result		0	No	3
Illegal code	Illegal contents				Actual
	“Upper order name #duetime” should be unique in the upper order name and this event.				

2.13. Definition of Basic Element Data

2.13.1. Customer Data（customer）

The information on customer is defined. Customer is the issue agent to make an order. Customer finally consumes item (stock) and resource (load).

Tag name	customer		Implementation level		1
Upper tag	pslx(1), parameter(1),				
Reference	order(1)				
Attribute	Type	Constraint	Required	Omitted	Actual

name	ID	Reference key	○		1
ac	string	create | delete | revise			1
calculation	string	sum | ave | cnt | min | max | earliest | latest | shortest | longest			3
Element name	Explanation		min	max	Actual
query	How to inquire customer		0	No	1
description	Optional note to customer		0	1	1
display	How to display customer		0	1	2
priority	Priority of customer		0	1	1
spec	Specification optionally established		0	No	1
address	Location of customer		0	1	1
Illegal code	Illegal contents				Actual
No					

2.13.2. Supplier Data（supplier）

The information on supplier is defined. Supplier is the final order destination and the supply origin of item (stock) and resource (load).

Tag name	supplier		Implementation level		1
Upper tag	pslx(1), parameter(1),				
Reference	order(2)				
Attribute	Type	Constraint	Required	Omitted	Actual
name	ID	Reference key	○		1
ac	string	create | delete | revise			1
calculation	string	sum | ave | cnt | min | max | earliest | latest |			3

		shortest | longest			
Element name	Explanation		min	max	Actual
query	How to inquire supplier		0	No	1
description	Optional note to supplier		0	1	1
display	How to display supplier		0	1	2
priority	Priority of supplier		0	1	2
spec	Specification established for supplier		0	No	1
address	Location of supplier		0	1	1
Illegal code	Illegal contents				Actual
No					

2.13.3. Item Data（item）

The information on item is defined. Item is produced or consumed by operation. To put it concretely, product, sub assembly, work in process, parts, raw material, and material correspond to item.

Tag name	item		Implementation level		1
Upper tag	pslx(1), parameter(1),				
Reference	item(2), order(1), produce(1), consume(1), partof(2), condition(2), action(3), lot(1),				
Attribute	Type	Constraint	Required	Omitted	Actual
name	ID	Reference key	○		1
Ac	string	create | delete | revise			1
calculation	string	sum | ave | cnt | min | max | earliest | latest | shortest | longest			3
master	boolean			false	2
parent	IDREF	Referring to item key			2

Element name	Explanation	min	max	Actual
query	How to inquire item	0	No	1
description	Optional note to item	0	1	1
display	How to display item	0	1	2
priority	Priority of item	0	1	2
spec	Specification established for item	0	No	1
price	Price of item (unit price)	0	1	1
calendar	Calendar of the available item to be supplied	0	No	3
capacity	Amount of the available items to be supplied	0	No	2
stock	Stock level of item	0	No	1
location	Location data of item	0	No	3
partof	Inclusion relation of item (included item)	0	No	2
lotsize	Lot size constraint of item	0	No	3
interval	Existence time constraint of item	0	No	3
Illegal code	Illegal contents			Actual
	The reference destination of parent must be master.			
	Calendar elements must be sorted in order of early value of time attribute.			
	Stock elements must be sorted in order of early value of time attribute.			

2.13.4. Resource Data（resource）

The information on resource is defined. Resource is used for a fixed period of time by operation. The operation load during that period is handled by the available production capacity to be supplied. To put it concretely, machine, facility, device, tool, labor, line and so on are resources.

Tag name	resource		Implementation level		1
Upper tag	pslx(1), parameter(1),				
Reference	resource(2),, event(1), assign(1), partof(2), condition(2), action(3), task(1), order(3)				
Attribute	Type	Constraint	Required	Omitted	Actual
name	ID	Reference key	○		1
ac	string	create | delete | revise			1
calculation	string	sum | ave | cnt | min | max | earliest | latest | shortest | longest			3
master	boolean			false	2
parent	IDREF	Referring to resource key			2
type	string	make | stock | move | check		make	3
Element name	Explanation		min	max	Actual
query	How to inquire resource		0	No	1
description	Optional note to resource		0	1	1
display	How to display resource		0	1	2
priority	Priority of resource		0	1	2
spec	Specification established for resource		0	No	1
price	Price of resource use（unit price）		0	1	2
calendar	Calendar data of resource		0	No	3
capacity	Capacity data of resource		0	No	2
load	Sum of resource loads in every date and time		0	No	1
location	Location of resource		0	No	3

partof	Inclusion relation of resource	0	No	2
tasksize	Proper task size of resource	0	No	3
changeover	Setup time for resource	0	No	2
Illegal code	Illegal contents			Actual
	The reference destination of parent must be master.			
	Calendar elements must be sorted in order of early value of time attribute.			
	Load elements must be sorted in order of early value of time attribute.			

2.13.5. Operation Data（operation）

The information on operation is defined. Operation is the action to consume and produce any item. In such a case, the specific resource is required. This element can be used as operation master information or individual schedule.

Tag name	operation		Implementation level		1
Upper tag	pslx(1), parameter(1),				
Reference	operation(2), order(2), predecessor(1), successor(2), partof(2), changeover(3), interval(3),				
Attribute	Type	Constraint	Required	Omitted	Actual
name	ID	Reference key	○		1
ac	string	create | delete | revise			1
calculation	string	sum | ave | cnt | min | max | earliest | latest | shortest | longest			3
master	boolean			false	2
parent	IDREF	Referring to			2

		operation key			
order	IDREF	Referring to order key			1
direction	string	forward | backward		forward	1
type	string	make | stock | move | check		make	3
Element name	Explanation		min	max	Actual
query	How to inquire operation		0	No	1
description	Optional note to operation		0	1	1
display	How to display operation		0	1	2
priority	Priority of operation		0	1	1
spec	Specification established for operation		0	No	1
price	Price of operation		0	1	2
start	Start event of operation		0	1	1
end	End event of operation		0	1	1
ev	Event other than start and end		0	No	2
qty	Operation size coefficient		0	1	2
duration	Operation processing time		0	1	1
progress	Operation progress		0	No	1
predecessor	Earlier operation than the operation		0	No	1
successor	Later operation than the operation		0	No	2
assign	Resource required for operating		0	No	1
produce	Item produced by operation		0	No	1

consume	Item consumed by operation	0	No	1
partof	Inclusion relation of operation (included operation)	0	No	2
lotsize	Constraint of lot size	0	No	3
tasksize	Constraint of task size	0	No	3
Illegal code	Illegal contents			Actual
	The reference destination of parent must be master.			

2.13.6. Order Data（order）

Orders are a sales order from customer and a purchase order to supplier besides manufacturing order directly related to actual manufacturing. This tag expresses the order information like the above orders.

Tag name	order		Implementation level		1
Upper tag	pslx(1), parameter(1),				
Reference	partof(2), pegging(2), operation(1), order(2), lot(1), task(1),				
Attribute	Type	Constraint	Required	Omitted	Actual
name	ID	Reference key	○		1
ac	string	create | delete | revise			1
calculation	string	sum | ave | cnt | min | max | earliest | latest | shortest | longest			3
master	boolean			false	2
parent	IDREF	Referring to order key			2
customer	IDREF	Referring to customer key			1

supplier	IDREF	Referring to supplier key			2
item	IDREF	Referring to item key			1
resource	IDREF	Referring to resource key			3
operation	IDREF	Referring to operation key			2
status	string	fixed | forcast | unofficial | temporal | done | cancel		fixed	1
direction	string	forward | backward		forward	1
multiplier	IDREF	Referring to unit key			3
Element name	Explanation		min	max	Actual
query	How to inquire order		0	No	1
description	Optional note to order		0	1	1
display	How to display order		0	1	2
priority	Priority of order		0	1	1
spec	Specification established for order		0	No	1
price	Price of order （total）		0	1	1
qty	Quantity of orders		0	1	1
release	Earliest start date of order		0	1	1
duetime	Latest end date of order （order duetime）		0	1	1
progress	Order progress		0	No	1
partof	Inclusion relation of order （upper class order）		0	No	2
pegging	Pegging relation of order		0	No	2
lotsize	Constraint of lot size		0	No	3
tasksize	Constraint of task size		0	No	3

Illegal code	Illegal contents	Actual
	The reference destination of parent must be master.	
	Either customer attribute or supplier attribute should be specified.	
	One of item attribute, operation attribute, and resource attribute should be specified.	
	When specifying lotsize, item must be selected. When specifying tasksize, resource must be selected.	
	When specifying multiplier, resource must be selected.	
	Multiplier must be a unit of time.	

2.13.7. Lot Data（lot）

Lot expresses that one item is produced or consumed by order. Ordinarily the meaning of lot is the same as individual item to be produced or consumed.

Tag name	lot		Implementation level		1
Upper tag	pslx(1), parameter(1),				
Reference	produce(1), consume(1), tracking(2),				
Attribute	Type	Constraint	Required	Omitted	Actual
name	ID	Reference key	○		1
ac	string	create | delete | revise			1
calculation	string	sum | ave | cnt | min | max | earliest | latest | shortest | longest			3
item	IDREF	Referring to item key			1
order	IDREF	Referring to			1

		order key			
type	string	produce | consume		produce	2
Element name	Explanation		min	max	Actual
query	How to inquire lot		0	No	1
description	Optional note to lot		0	1	1
display	How to display lot		0	1	2
priority	Priority of lot		0	1	1
spec	Specification established for lot		0	No	1
price	Price of lot		0	1	2
qty	Quantity of lots		1	1	1
time	Lot production (consumption) date		0	1	2
address	Location of lot		0	1	3
tracking	Correspondence of lot		0	No	2
Illegal code	Illegal contents				Actual
No					

2.13.8. Task Data（task）

Task shows that one operation concretely uses resource by order. From the viewpoint of resource, the sum of individual tasks is calculated as a load of the resource.

Tag name	task		Implementation level		1
Upper tag	pslx(1), parameter(1),				
Reference	assign(1), tracking(3),				
Attribute	Type	Constraint	Required	Omitted	Actual
name	ID	Reference key	○		1
ac	string	create | delete | revise			1
calculation	string	sum | ave | cnt | min | max | earliest |			3

		latest | shortest | longest			
resource	IDREF	Referring to resource key			1
order	IDREF	Referring to order key			1
type	string	produce | consume		produce	3
multiplier	IDREF	Referring to unit key			3
Element name	Explanation		min	max	Actual
query	How to inquire task		0	No	1
description	Optional note to task		0	1	1
display	How to display task		0	1	2
priority	Priority of task		0	1	2
spec	Specification established for task		0	No	1
price	Price of task		0	1	2
qty	Quantity of task loads		1	1	1
time	Occurrence date and time of task		0	1	2
address	Occurrence address of task		0	1	3
tracking	Correspondence of task		0	No	3
Illegal code	Illegal contents				Actual
	Multiplier must be a unit of time.				

2.14. Definition of Data Related with Planning

2.14.1. Expression Data（expression）

A numerical formula consisting of multiple members is expressed. And also the relation configuration such as constraint or the aim concept

such as evaluation can be represented. Moreover a mathematical model can be expressed with combining these expressions.

Tag name	expression		Implementation level		1
Upper tag	pslx(1), op(3)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
name	ID	Reference key	○		1
ac	string	create | delete | revise			1
rhd	double			0	2
value	double			0	1
violation	double			1	2
unit	IDREF	Referring to unit			2
type	string	EQ | NE | GT | GE | LT | LE |MAX | MIN		EQ	2
Element name	Explanation		min	max	Actual
description	Note		0	1	1
display	How to display		0	0	3
priority	Priority		0	0	2
spec	Various specifications		0	No	2
op	Elements making an expression		0	No	1
Illegal code	Illegal contents				Actual
	The reference of expression through op should not be a loop.				

2.14.2. Operation Division（op）

The operation division of members of numerical expression is expressed. Individual members are combined with arithmetic operators like ＋, －, ＊, ／ and so on.

Tag name	op		Implementation level		1
Upper tag	expression(1),				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
no	long	Serial integers from 1		1	1
operator	string	+ | - | * | /		+	1
value	double				1
parameter	IDREF	Referring to parameter			1
expression	IDREF	Referring to expression			3
Element name	Explanation		min	max	Actual
No					
Illegal code	Illegal contents				Actual
	The reference of expression should not be a loop.				
	Only one of parameter, expression and value can be set up.				

2.14.3. Parameter（parameter）

This data indicates various constants, variables or calculation values. The subject for basic elements such as item, resource, order, and operation is specified when the data is a calculation. Calculation is executed by giving the retrieval information such as period information.

Tag name	parameter		Implementation level		1
Upper tag	pslx(1), op(1)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
name	ID	Reference key	○		1
ac	string	create | delete | revise			1
value	double				

unit	IDREF	Referring to unit			2
index	long			0	3
type	string	variable | constant		variable	1
Element name	Explanation		min	max	Actual
description	Note		0	1	1
display	How to display		0	1	3
priority	Priority		0	1	2
spec	Various specifications		0	No	2
customer	Intended customer data		0	No	1
supplier	Intended supplier data				1
item	Intended item data				1
resource	Intended resource data				1
lot	Intended lot data				1
task	Intended task data				1
event	Intended event data				1
operation	Intended operation data				1
order	Intended order data				1
Illegal code	Illegal contents				Actual
	When type attribute is constant, the element that is item or below may not be set up.				

2.15. Information for Querying Data

2.15.1. Query Data（query）

This is the tag for querying PSLX data. When a part of PSLX data is intended, selecting the element to be inquired and a partial subject can be specified.

Tag name	query	Implementation level	1
Upper tag	item(1), resource(1), lot(1), task(1), customer(1), supplier(1), event(1), operation(1), order(1)		
Reference			

Attribute	Type	Constraint	Required	Omitted	Actual
select	string				1
depth	int			0	3
start	datetime				2
end	datetime				2
calculation	string	start | end | sum | ave | min | max | cnt		end	2
Element name	Explanation		min	max	Actual
No					
Illegal code	Illegal contents				Actual
	The value of select attribute must be one of sub-element name of the parent element in which this element is set up or what is approved as reverse reference attribute, or all, attribute.				
	End date must be later than start date.				
	Depth attribute can be set up only when select attribute is tracking, pegging, partof, predecessor, or successor.				

2.15.2. Specifying The Minimum Value（min）

The minimum value of numerical data is set up. This data is used for specifying the range when inquiring data.

Tag name	min		Implementation level		2
Upper tag	qty(2), price(2), priority(2)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
value	double		○		2
exclusive	boolean			false	3
Element name	Explanation		min	max	Actual
No					

Illegal code	Illegal contents	Actual
No		

2.15.3. Specifying The Maximum Value（max）

The maximum value of numerical data is set up. This data is used for specifying the range when inquiring data.

Tag name	max		Implementation level		2
Upper tag	qty(2), price(2), priority(2)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
value	double		○		2
exclusive	boolean			false	3
Element name	Explanation		min	max	Actual
No					
Illegal code	Illegal contents				Actual
No					

2.15.4. Specifying The Earliest Date（earliest）

The earliest value of date data is set up. This data is used for specifying the range when inquiring data.

Tag name	earliest		Implementation level		2
Upper tag	time(2)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
value	datetime		○		2
exclusive	boolean			false	3
Element name	Explanation		min	max	Actual
No					
Illegal code	Illegal contents				Actual

No		

2.15.5. Specifying The Latest Date（latest）

The latest value of date data is set up. This data is used for specifying the range when inquiring data.

Tag name	latest		Implementation level		2
Upper tag	time(2)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
value	datetime		○		2
exclusive	boolean			false	3
Element name	Explanation		Min	max	Actual
No					
Illegal code	Illegal contents				Actual
No					

2.15.6. Specifying The Shortest Time（shortest）

The shortest value of time data is set up. This data is used for specifying the range when inquiring data.

Tag name	shortest		Implementation level		2
Upper tag	duration(2)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
value	duration		○		2
exclusive	boolean			false	3
Element name	Explanation		min	max	Actual
No					
Illegal code	Illegal contents				Actual
No					

2.15.7. Specifying The Longest Time（longest）

The longest value of time data is set up. This data is used for specifying the range when inquiring data.

Tag name	longest		Implementation level		2
Upper tag	duration(2)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
Value	duration		○		2
exclusive	boolean			false	3
Element name	Explanation		min	max	Actual
No					
Illegal code	Illegal contents				Actual
No					

2.15.8. Proposed Character Data（enumerate）

This data is used for enumerating the proposed character data. This data is used for specifying the range when inquiring data.

Tag name	enumerate		Implementation level		3
Upper tag	char(3)				
Reference					
Attribute	Type	Constraint	Required	Omitted	Actual
value	string		○		3
Element name	Explanation		min	max	Actual
No					
Illegal code	Illegal contents				Actual
No					

3. How to Describe XML Tag

3.1. Basic Elements and Configuration

The content of message by PSLX is expressed by defining the relations and attributes of the below basic elements.

Customer・Supplier　Customer is the order origin and supplier is the last order destination. The former process and the later process can be taken as a customer or a supplier besides business partner and trader outside the enterprise. Customer is expressed with a <customer> tag and supplier is expressed with a <supplier> tag.

Order　Order is the basic information to generate any plan or action. Manufacturing order and operation order exist in the various situations in production field besides a sales order and a purchase order. Order is expressed with a <order> tag.

Operation　Operation is the action to consume any item and to produce any item in exchange for consuming the item. It is necessary to keep the resources required for actually operating for the fixed period. Operation is expressed with a <operation> tag.

Event　Event is the action to change the target world and one of execution date can be selected out of real time. Start and end are also one kind of events, however there are events independent of operation. Event is expressed with a <event> tag.

Item　Item is consumed or produced by operation. Product, parts, material, work in process can be set as an item. Item has the amount of stocks in the time series. Item is expressed with a <item> tag.

Resource	Resource is required for operating and the capacity is set beforehand. Some operations can be executed at the same time within the capacity of resource. Device, machine, facility, labor, and tool are given as resources. Resource is expressed with a <resource> tag.
Lot	Lot indicates the lump of one item produced or consumed by one operation. The quantity of stocks can be calculated by adding up lots in every date. Lot is expressed with a <lot> tag.
Task	Task indicates the unit that one resource is used by operation. When some resources execute one operation, the number of tasks corresponds to the number of the resources. Task is expressed with a <task> tag.

In XML schema of PSLX, the targets are expressed by defining the attribute and the extension constraint for these basic elements and by specifying the value attribute and the character attribute at more detailed level.

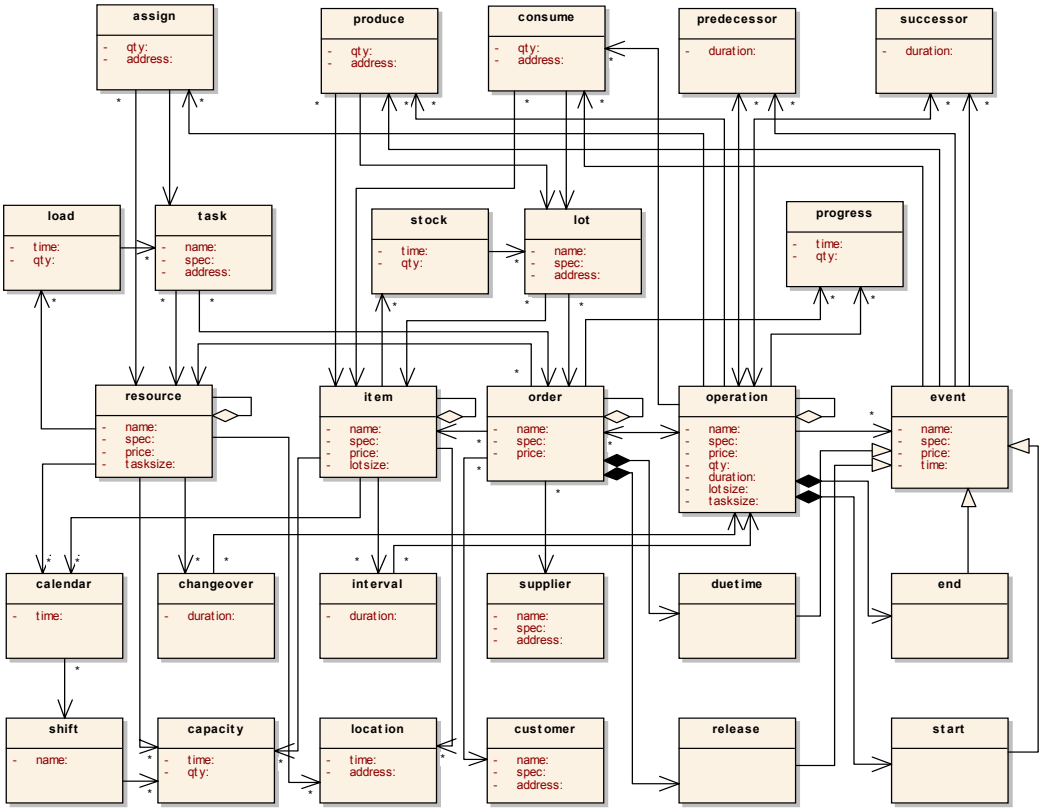


Figure　XML Tag Configuration in PSLX

3.2. Top Level and Profile

3.2.1. Top-level Information

<pslx> tag is used on the top level of PSLX. Type attribute, id attribute, ref attribute, action attribute and receipt attribute are set in a <pslx> tag.

Type attribute shows the situation where the PSLX data is used. The value of type attribute is selected out of the following items. When the value is omitted, the data is regarded as a file.

Attribute value	Explanation
request	Show that PSLX data is a request message.
response	Show that PSLX data is a response message.
receipt	Shows that PSLX data is a receipt message.
exception	Show that PSLX data is an exception message.

file	Show that PSLX data is used for making permanent.

The ID (character string) that makes PSLX message unique in the application creating a message is set in id attribute. This attribute must be set when type attribute is the value except file.

Ref attribute is used when PSLX data refers to the other PSLX data. This attribute must be set in response message, receipt message and exception message. ID of request message is set in a response message and ID of the applicable request message or response message is set in a receipt message and an exception message.

The interface name defined in PSLX is set in action attribute. This attribute must be set in the case where type attribute is the value other than file. The same value as the value of action attriute in the message referred to by ref attribute must be set in action attributes of response message, receipt message and exception message.

Receipt attribute is used for confirming that a message is received. This attribute can be set only in a request message and a response message. The application receiving the request message or the response message with the receipt attribute, true must return a receipt message to the sender immediately.

<error> tag can be set in the top level. <error> tag is used for notifying an error or warning information. Severity attribute distinguishes between error and warning. “Error” in the severity attribute indicates error, and “warning” indicates warning.

The error information can be set only in an exception message. The warning information can be set only in an exception message, a receipt message and a response message.

The following shows the samples of error and warning set in the exception message.

```
<pslx type=“exception” ref=“a001”>
<error code=“E02004” severity=“error”>the applicable ID cannot be found.</error>
<error code=“E02015” severity=“error”>value setting error</error>
<error code=“W08002” severity=“warning”>the attribute value is ignored.</error>
```

```
</pslx>
```

<spec> tag on the top level can be used for setting the information added to each message separately. This is used for individually extending the function of interface in each application.

3.2.2. Profile Information

<profile> tag which is the top-level element is used for setting the information to the whole PSLX data. Name attribute, author attribute, version attribute, create attribute, expire attribute, current attribute, start attribute and end attribute can be set in this tag.

The target problem name is set in name attribute. When making PSLX data permanent, the data must be unique by this value and version attribute.

When PSLX data is modified for the same problem, the version is managed and identified by version attribute. The value of version attribute is the optional character string and the content of the character string can be set freely.

Create attribute expresses the date when the PSLX data is valid, and expire attribute expresses the date when the PSLX data is invalid. If the date when the application executes processing is out of this range, the PSLX data may not be used.

Start attribute, end attribute and current attribute show start date, end date and present date of the target period for PSLX data. The start date, end date and present date that are set in each attribute must be able to be referred to by #init, #final and #now which are the date constants.

3.3. Expressing Numerical Information

<qty> tag can express many numerical values used in PSLX data. <char> tag can express many character strings. Unit system, fraction information and constraint information can be specified using these tags for setting each value.

3.3.1. Expressing Value

<qty> tag expresses the numerical information. And <char> tag expresses the character information.

For example, a numerical value, 0.5 is expressed as below.

```
<qty value="0.5"/>
```

A character string, "clear, with some clouds" is expressed as below.

```
<char value="clear with some clouds"/>
```

Price and priority are special cases of the numerical information. Price is expressed with a <price> tag. Priority is expressed with a <priority> tag.

For example, the following shows the price is 150 yen.

```
<price value="150" unit="yen"/>
```

The priority is expressed with the numbers from –1 to 1. For example, the following shows the maximum priority.

```
<priority value="1.0"/>
```

Address is a special type of character string information. This is used for specifying a spatial location. Address is expressed with a <address> tag. Address can be specified by URI attribute, east, south, and height attribute besides value attribute.

```
<address value="3-7-2, Kajino-cho, Koganei City"/>
<address URI="www.pslx.org/demo/001"/>
<address east="200" south="100" height="3"/>
```

The information can be added to address with a tag of <descripton>, <display>, <priority> or <spec>.

Description information is for setting an optional character string in a basic element as a description. Description information is expressed with a <description> tag. This is a kind of character information but different from other character information. The content is set in the tag itself, not in value attribute.

```
<description>
An optional note can be set here.
</description>
```

3.3.2. Expressing Fraction

<qty> tag and <price> tag indicating numerical values can express a fraction by using a <base> tag. And also this <base> tag is available to <translate> used for unit translation. This function must be handled at implementation level 2 and over. <base> tag may not be used for the application at implementation level 1.

For example, the processing time per one product in case that it takes ten minutes to produce three products is expressed as below.

```
<qty value="10" unit="min">
<base value="3"/>
</qty>
```

3.3.3. Expressing Date

Date is expressed with a <time> tag. There are two ways to set the value of date. The date is directly specified using value attribute. A numerical value and scale are specified using count attribute and scale attribute and are converted inside.

For example, the date and time, 12:30 on April 10, 2003 is directly described as a value of value attribute as below.

```
<time value="2003-04-10T12:30:00.000+9.00"/>
```

The following example shows the date and time of one day later from the standard date set in the scale information.

```
<time count="1" scale="day"/>
```

For example, if the scale "day" separately specified is set as below, the above example shows 0:00 a.m. on April 11, 2003.

```
<scale name="day" unit="#day" origin="2003-04-10T00:00:00.000+9.00"/>
```

3.3.4. Date Constant

Date constants are three kinds: #now indicating now date and time, #init indicating the start date and time of the planning period, and

#final indicating the final date and time of the planning period. These date constants can be referred to by ref attribute in a <time> tag.

The values specified with current attribute, start attribute and end attribute in a <profile> tag are set in date constants, #now, #init and #final respectively. If these data aren't set in a <profile> tag, the application must set the value independently.

The following shows that the stock value at the start point of planning period is 100.

```
<stock>
<time ref="#init"/><qty value="100"/>
</stock>
```

3.3.5. Expressing Time

Time information is expressed with a <duration> tag. The value of time is ISO8601 format and expressed like PnYnMnDTnHnMnS. One hour is PT1H and expressed as below.

```
<duration value="PT1H"/>
```

The unit can be specified in time with unit attribute. In such a case, count attribute is used, not value attribute. For example, the following example shows the same content as the above example because the time is a unit.

```
<duration count="1" unit="#hour"/>
```

The time units reserved beforehand are #sec, #min, #hour, #day, #week, #10d, #15d, #month and #year. #10d is the unit that divides a month into three parts: the beginning, the middle, and the end. #15d is the unit that divides a month into two parts: the beginning and the end.

If the unit other than the above reserved units is used, the original unit can be set and used.

In the following case, the unit, "half-hour" showing that one unit is 30-minute is defined and the date and time are expressed using it.

```
<unit name="half-hour"><translate unit="#hour" value="0.5"/></unit>
<duration count="2" unit="half-hour"/>
```

3.4. Handling Unit System

3.4.1. Setting and Converting Unit System

The units in <qty>, <base>, <lotsize>, <tasksize> and <price> showing numerical values can be specified by unit attribute. The contents of <translate> and <scale> can be defined with unit attribute.

The value specified with unit attribute is the reserved unit or the unit system defined with a <unit> tag. The reserved unit systems are #sec, #min, #hour, #day, #week, #10d, #15d, #month and #year.

If the unit system is defined originally, <unit> tag is used. The defined unit system can be specified as a value of unit attribute like the defined unit system. However "#" may not be used for the head of name of the unit system to be newly defined. The function using <base> tag related with unit system must be handled at implementation level 2 and over.

The unit systems originally defined are as below.

```
<unit name="piece"/>
<unit name="kg"/>
<unit name="p/c"/>
<unit name="yen"/>
```

The numerical information is set using these unit systems as below.

```
<qty value="10" unit="piece"/>
<qty value="120" unit="min"/>
<price value="3000" unit="yen"/>
```

The fraction type can also be set as a unit system. The unit system is changed into fraction type by using a <base> tag and defining the unit attribute under the tag.

The following case means that the price of operation is 50 yen per one piece according to the produced lot.

```
<operation name="P001">
<price value="50" unit="yen"><base unit="piece"/></price>
</operation>
```

When the value is automatically converted between some unit systems, the way of converting unit is set. The way of converting unit is expressed with a <translate> tag. This function must be handled at implementation level 3.

At the time when the unit is defined, a <translate> tag is added to a <unit> tag for defining the unit. The quantity and unit defined with the <translate> tag correspond to one unit of <unit>, the upper element.

In other words, if the unit defined with a <unit> tag is X; the unit defined with a <translate> tag is Y; the translation coefficient defined with a <translate> tag is a; the original point is z, value x of unit system X is translated into the unit system Y by A・x＋z.

For example, the relation between unit systems "piece" and "kg" for the product that is 5 kg in the box with 0.5 kg weight is expressed as below.

```
<unit name="piece">
<translate value="5" unit="kg" origin="0.5"/>
</unit>
```

Unit translation is reversible and so when one way of translating is defined, the reverse translation must be available. For instance, the above example means that 1 kg corresponds to 1/5 piece in the rest taking 0.5 kg from the whole weight

3.4.2. Using Scale

The date and the time information can be corrected by the scale information set with a <scale> tag. It is possible to say that <scale> is the special form in which <unit> and <translate> are replaced in order to use <scale> especial for date or time.

Unit attribute, value attribute and origin attribute are used for setting the display scale. Unit attribute and value attribute indicate the time corresponding to one unit. If unit attribute is omitted, #min is set.

When the value of name attribute is #display for defining scale, the

default information of display scale can be changed.

The example of scale definition is shown. For example,

```
<scale name="aaa" unit="#min" value="10" origin="2003-04-08T00:00:00"/>
```

under the above scale, the following two expressions indicate the complete same date and time.

```
<time count="20" scale="aaa"/>
<time value="2003-04-08T03:20:00"/>
```

The followig calculation is executed inside.

```
t = origin + unit (count×value)
  ="2003-04-08T00:00:00"+ (20×10) min
  ="2003-04-08T03:20:00"
```

When origin attribute is omitted, #init date which is a date constant is selected instead of the attribute.

3.4.3. Expressing Discrete Time (Period)

<time> tag expressing the date may express the usual continuous time and the period information such as a time packet. When type attribute of scale tag <scale> is "disc" to the value of scale attribute set in the date and time, the tag exresses the discrete time.

The abbreviation form of this attribute is "cont." This "cont" indicates the date and time in the flow of usual continuous time, while the discrete time is the period (section) information expressed by time packet or time slice.

The date and time in a discrete system are expressed with count attribute and with the integers that the standard date is 0. For example,

```
<scale name="bbb" unit="#day" value="1" origin="2003-04-08T00.00.00"
type="disc"/>
```

under the above scale,

```
<time count="0" scale="bbb"/>
```

the above information shows the period of one day (24 hours) from 0:00 a.m. on April 8, 2003 to the time just before 0:00 a.m. on the next day. Even if count is used for specifying a <time> tag, the tag indicates the series date and time in case that type attribute of <scale> is ”cont.” For instance, the above case indicates 0:00 a.m. on April 8, 2003.

3.4.4. Setting Display Scale

When displaying various elements on the display device like a screen, the scale to display is specified on the top level with a <scale> tag and the scale is referred to in a <display> tag with scale attribute. When scale attribute is omitted in a <display> tag, #display is set. When using some display scales at the same time, the display scales can be put to proper use with a name set at the time when each scale is defined.

The following shows that the original display point on the screen (the left end in the display area) is the date, 0:00 a.m. on 2003/04/08 and one division of a scale indicates 10 minitues.

```
<scale name=”scale1” unit=“min” value=“10” origin=“2003-04-08T00.00.00”/>
```

3.5. Expressing Time Series Information

Stock value, load value and capacity value change with time. These data are expressed in the form of a pair of the value and the date information to each attribute as the time series information.

The general form of time series information is a <spec> tag. Usually the specification is the fixed information but the <spec> tag in PSLX can express the ever-changing information using value attribute and time element.

The following shows that the temperature of product A is 10 degrees at 9:00 a.m., and 25 degrees at 3:00 p.m.

```
<item name=“product A”>
<spec name=“temperature”><time value=“2003-04-10T09:00:00”/><qty
value=“10” unit=“degree”/></spec>
<spec name=“temperature”><time value=“2003-04-10T15:00:00”/><qty
value=“25” unit=“degree”/></spec>
</item>
```

Some elements with the same name attribute can be defined in a <spec> tag by changing the date and time. However each element must be defined in order of early date.

When there are some elements with the same specification name and the same date and time, the relative attribute of the second and later data must be true.

If a date tag <time> isn't attached, the value set with <spec> is regarded as being fixed for planning period.

```
<spec name="specification A"><qty value="100"/></spec>
```

The above description has the same value as the following description.

```
<spec name="specification A"><qty value="100"/><time ref="#init"/></spec>
```

If the date and time are specified with a date tag <time>, it means that the specification value is the value specified with value attribute at the date and time. When type attribute is "disc," the value is continued in future. When type attribute is "cont," the value continuously changes to the value to be set next.

The value specified with <spec> changes like steps in each specified date as the following (left-side) figure when type attribute is "disc". When type attribute is "cont," the value continuously changes to the value to be defined next as the following figure.

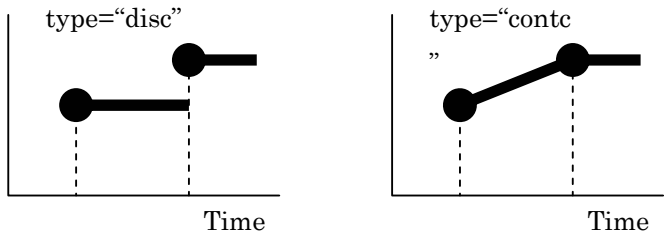


Figure Discrete Change and Continuous Change

The set value is fixed from the date specified with <time ref="#init"> to the next date and from the last defined date and later in spite of type

attribute.

The specification value may be a character string, not a numerical value. For example, the following shows that the condition of machine A is abnormal at 14:00.

```
<resource name="machine A">
<spec name="status"><time value="2003-04-10T14:00:00"/>
<char value="abnormal"/></spec>
</resource>
```

3.5.1. Setting Stock and Load

<stock> tag showing the stock value is the special type of specification tag <spec>. The stock value is the information that is an element of <item> tag indicating item and expresses the sum of stock values of the item in time series.

For example, the following shows that the stock value of product A is 150 on April 10 and 130 on April 11.

```
<item name="product A">
<stock><time value="2003-04-10T00:00:00"/><qty value="150"/></stock>
<stock><time value="2003-04-11T00:00:00"/><qty value="130"/></stock>
</item>
```

The stock value can be set with a relative value by making relative attribute "true" besides with an absolute value. When setting the stock value with a relative value, only the change amount of stock values at the date is specified.

If the condition is the same as the above example and the stock value of product A is 150 on April 10 and 130 on April 11, the following example shows that the stock value is set with a relative value.

```
<item name="product A">
<stock relative="true"><time value="2003-04-10T00:00:00"/><qty
value="150"/></stock>
<stock relative="true"><time value="2003-04-11T00:00:00"/><qty
value="-20"/></stock>
</item>
```

Some lot data that are the causes of change can be specified with a <lot> tag in a <stock> tag of the stock value.

3.5.2. Setting Load Value

<load> tag indicating the load value can be set to resource <resource>. This indicates the sum of used resource (task) at the specific date.

The following example shows that two labors as resources are required from 9:00 to 14:00 and five labors are required to 16:00.

```
<resource name="labor">
<load><time value="2003-04-19T09:00:00"/><qty value="2"
unit="person"/></load>
<load><time value="2003-04-19T14:00:00"/><qty value="5"
unit="person"/></load>
<load><time value="2003-04-19T16:00:00"/><qty value="0"
unit="person"/></load>
</resource>
```

The load value can be specified with a relative value, not an absolute value using relative attribute like the stock value. When expressing the former example with a relative value, the value is rewritten as below.

```
<resource name="labor">
<load relative="true"><time value="2003-04-19T09:00:00"/>
  <qty value="2" unit="person"/></load>
<load relative="true"><time value="2003-04-19T14:00:00"/>
  <qty value="3" unit="person"/></load>
<load relative="true"><time value="2003-04-19T16:00:00"/>
  <qty value="-5" unit="person"/></load>
</resource>
```

Some task data that are the causes of the change can be specified with a <task> tag in a <load> tag of the load value.

3.5.3. Operation Progress

Progress of operation and order can be expressed with a <progress> tag. <progress> tag is set in the operation or order in which progress is set and the measuring date and the content (quantity or status) are described in the tag. Some <progress> tags can be set in every date to express the status where the progress changes in time series.

The following example shows that operation "P001" is completed at 0% at 18:00 on April 10, 80% on 11, and 100% on 12.

```
<operation name="P001">
<progress><time value="2003-04-10T18:00:00"/><qty value="0.0"/></progress>
<progress><time value="2003-04-11T18:00:00"/><qty value="0.8"/></progress>
<progress><time value="2003-04-12T18:00:00"/><qty value="1.0"/></progress>
```

```
</operation>
```

The status as the progress information can be expressed using status attribute with the following idetification characters.

Attribute value	Explanation
completed	Status where operation or order is finished completely.
canceled	Status where operation or order is canceled halfway.
suspended	Status where operation or order is suspended temporarily.
started	Status where operation or order is being executed.
ready	Status where operation or order can be executed but not executed yet.
created	Status where operation or order cannot be executed yet.

The following shows that order ”K001” is accepted on April 3 and ready to start from 5, started from 7 and completed on 9.

```
<order name=“K001”>
<progress status=“created”><time value=“2003-04-03T18:00:00”/></progress>
<progress status =“ready”><time value=“2003-04-05T18:00:00”/></progress>
<progress status =“started”><time value=“2003-04-07T18:00:00”/></progress>
<progress status =“completed”><time value=“2003-04-09T18:00:00”/></progress>
</order>
```

The following figure shows the status transition between statuses expressed by progress <progress>.

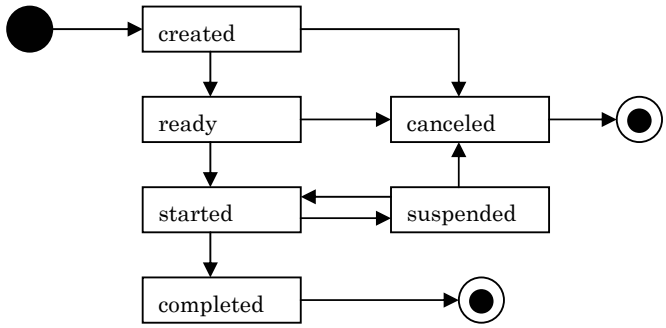


Figure　Order Status Transition

3.5.4. Usage of Loaction Information

<item> tag indicating item and <resource> tag indicating resource have the location information. However these locations are set with a <location> tag, not <address> indicating a static location because they may change with time.

<location> tag indicating one dynamic data has a pair of a date <time> tag and a location <address> tag as elements.

The following example means the situation that the resource, transportation car is at place A at 13:00 and at place B at 16:00.

```
<resource name="transportation car">
<location><time value="2003-04-10T13:00:00"/><address name="place A"/></location>
<location><time value="2003-04-10T16:00:00"/><address name="place B"/></location>
</resource>
```

3.6. Expressing Capacity and Calendar

3.6.1. Setting Capacity Information

The capacity value shows the upper and lower limits available to be actually set in the item or resource to the stock value set in item or the load value set in resource. The capacity value can be set with a <capacity> tag.

The following example shows that there are 5 labors from 8:00 to 12:00, 2 labors from 12:00 to 13:00, 7 labors from 13:00 to 17:00.

```
<resource name="labor">
<capacity><time value="2003-04-10T08:00:00"/><qty value="5" unit="person"/></capacity>
<capacity><time value="2003-04-10T12:00:00"/><qty value="2" unit="person"/></capacity>
<capacity><time value="2003-04-10T13:00:00"/><qty value="7" unit="person"/></capacity>
<capacity><time value="2003-04-10T17:00:00"/><qty value="0" unit="person"/></capacity>
</resource>
```

Setting capacity information is regarded as the upper limit setting unless otherwise specified. However when direction attribute is ”lower”, the lower limit information is set. Thus the safety stock value of item and the lower limit value of load value of resource can be

set.

3.6.2. Expressing Calendar Information

The calendar information is a kind of capacity information because the calendar information expresses whether the operation can be executed or not on each day. Therefore all the calendar information can be expressed by replacing with a <capacity> tag showing capacity information. However usually the calendar information is expressed using a <calendar> tag because it is easy to describe it.

First, the shift pattern information must be expressed with a <shift> tag in order to indicate the calendar information. The shift pattern information is the information that the specific capacity information is made as a pattern and available to be referred to at a unit of shift.

When registering the working time from 8:00 a.m. to 6:00 p.m. in every day as a pattern, it is expressed as below.

```
<shift name="day duty 8h">
<capacity><time value="2000-01-01T00:00:00"/><qty value="0"/></capacity>
<capacity><time value="2000-01-01T08:00:00"/><qty value="1"/></capacity>
<capacity><time value="2000-01-01T18:00:00"/><qty value="0"/></capacity>
</shift>
```

Calendar information is set by specifying a shift name and date to each resource with this shift pattern information.

One-week calendar of A factory can be expressed as below. "day duty 8h" is the shift pattern information defined before and moreover the shift pattern information such as "day duty 10h" or "day off" is defined beforehand.

```
<resource name="A factory">
<calendar shift="day duty 8h" time="2001-09-01T00:00:00"/>
<calendar shift="day duty 8h" time="2001-09-02T00:00:00"/>
<calendar shift="day duty 9h" time="2001-09-03T00:00:00"/>
<calendar shift="day duty 10h" time="2001-09-04T00:00:00"/>
<calendar shift="day duty 10h" time="2001-09-05T00:00:00"/>
<calendar shift="day off" time="2001-09-06T00:00:00"/>
<calendar shift="day off" time="2001-09-07T00:00:00"/>
</resource>
```

The capacity information of each resource can be produced on the application side by combining these calendar information and shift

pattern information.

In order to convert the calendar information and the shift pattern information into the capacity information, the value of time attribute in <calendar> is associated to the earliest date in a <capacity> tag in a shift pattern and when other <capacity> tags exist, the dates in the tags are shifted so that the relative date doesn't change on the shift pattern.

3.7. Expressing Basic Element

The basic elements making a planning problem or a scheduling problem are item, resource, lot, task, operation, event and order. Item is expressed with a <item> tag, resource with a <resource> tag, lot with a <lot> tag, task with a <task> tag, operation with a <operation> tag, event with a <event> tag and order with a <order> tag.

These basic element data always exist in the level just under a <pslx> tag and must be able to be referred to by an element ID. Each element in which ID is set can be referred to out of attributes of other elements.

When referring to element ID, the referred element information must not be defined in the same file or message unless the file or message is designated. As a matter of course, if the data corresponding to the referred element ID doesn't exist in the application receiving the message, it is an error.

The optional character string can be specified in an element ID but "#" may not be used as a head character.

3.7.1. Common Attribute

ID of basic element data is specified with name attribute.

Event, operation, item, resource, and order of the basic elements may be used as the master information and individual instance information. When the element is used as the master information, master attribute must be "true".

On the other hand, ID of the master information is specified in the value of parent attribute of the instance information produced based on master.

For example, the following means that the instance information, “cutting 001” is produced from the master “cutting”.

```
<operation name=“cutting” master=“true”/>
<operation name=“cutting 001” parent=“cutting”/>
```

On implementation level 2, the instance information, the basic element can take over the content of master information. And on implementation level 3, the master information can take over the same kind of other master information. In order to take over the content, ID of predecessor element must be specified in parent attribute for defining the basic element on the succeeding side.

The content of successor is produced as a template adding individual concrete values. The master information that is a template to the produced transaction information is set as the parent attribute.

Basic elements such as item, resource, operation, event and order have the following elements commonly.

<description> tag showing the description information can be used for setting the optional description information on a basic element.

<display> tag showing the display information indicates the individual way to display each basic element on the screen or document.

<priority> tag, which is the priority information, is used for setting the priority to a basic element. Priority is the parameter to be used in the logic of planning and scheduling.

The specification information, a <spec> tag is used for setting or inquiring the specification information after defining the optional specification name originally. The specification information can be expressed in time series if necessary.

3.7.2. Inclusion Relation

Inclusion relation of item, resource, operation and order can be expressed with a <partof> tag. In inclusion relation, a <partof> tag is set in the element equivalent to a part and the element equivalent to the whole is set in the tag. This function must be handled at implementation level 2 and over.

The following shows that product X is made from a set of product A and product B in case of item.

```
<item name="product X"/>
<item name="product A"><partof item="product X"/></item>
<item name="product B"><partof item="product X"/></item>
```

When setting the quantity of children to one unit of parent, the quantity can be expressed using a <qty> tag. The following example shows that 10 child parts are required for making one parent part.

```
<item name="child parts"><partof item="parent parts"><qty value="10"/></partof></item>
```

The example of inclusion relation of resource, which is the case where machine A is the element making line B, can be expressed as below.

```
<resource name="line B"/>
<resource name="machine A"><partof resource="line B"/>
</resource>
```

In case of operation, the inclusion relation of operation is expressed similarly. The inclusion relation can be used when one operation consists of some partial operations.

The following is the definition which setup A, processing B and transporting C are made up into one operation P and managed.

```
<operation name="operation P"/>
<operation name="setup A"><partof operation="operation P"/></operation>
<operation name="processing A"><partof operation="operation P"/></operation>
<operation name="transporting A"><partof operation="operation
  P"/></operation>
</operation>
```

When the inclusion relation of operation stands up, the start date of the operation indicating the whole must be the earliest date in the

start dates of all constituent operations and the end date must be the latest date in the end dates of all constituent operations.

The inclusion relation of order is used for managing individual order defined in every item as one order. For example, when ordering 10 of product A and 20 of product B, the content of order is expressed as below.

```
<order name="K001"/>
<order name="K001-1" item="product A">
<qty value="10" unit="piece"/><partof order="K001"/></order>
<order name="K001-2" item="product B">
<qty value="20" unit="piece"/><partof order="K001"/></order>
```

3.7.3. Expressing Action

Action can be defined as an action to change the time series information of basic elements, which are item and resource, such as increase and decrease of stock value or loads. Action is expressed with a <action> tag. This function must be handled at implementation level 3.

The information that can be changed by action is stock value, load value, capacity value, progress, location and the specification originally defined in a specification tag <spec>. Action tag, <action> can be set in a start tag <start>, an end tag <end>, an order duetime tag <duetime>, an order release tag <release> and other event tags <event>.

ID is set with item attribute for item and with resource attribute for resource as the element name set by action and the item to be set is specified with spec attribute. When the items to be set are stock value, load value, capacity value and progress, the value of spec attribute is #stoc, #load, #capacity and #progress respectively. When the specification is optional, the specification name is specified in spec attribute.

The following expresses the action to make the temperature of product X rise +5 degrees C.

```
<action item="product X" spec="temperature" relative="true">
<qty value="5" unit="digree C"/></action>
```

If relative attribute is omitted or ”false” is specified, the value just as specified with a <qty> tag is set.

Address and character string can be set in action besides numerical values. When specifying address, “#location” is specified in the value of spec attribute and the content is set with a <address> tag. If the numerical data, east, south and height are set in a <address> tag, relative attribute can be ”true”.

When setting a character string, not a numerical value to the defined specification, <char> can be used as below. The following example shows that the specification value, “status” of machine A is set at “off”.

```
<action resource=“machine A” spec=“status”><char value=“off”/></action>
```

The condition that the actions are executable at the same time can be set in the event setting action with a <condition> tag. The condition to execute <action> on the same level is that the evaluation of all the <condition> tags set under the same parent element is true.

The codition specifies the intended item or resource and a specification name and specifies the way to compare the specification value and the value set with char, qty or address by type attribute.

For example, the condition that the status of machine A is “ready” is expressed as below.

```
<condition resource=“machine A” spec=“status” type=“EQ”>
<char value=“ready”/></condition>
```

“EQ” meaning equal, “NE” meaning not-equal, “GT” meaning greater, ”LT” meaning more little, “GE” meaning equal or greater, and “LE” meaning equal or more little can be set in type attribute in the <condition> tag showing condition.

In a <changeover> tag indicating the switch relation and a <interval> tag indicating the exsistence period, the condition can be set with a <condition> tag. However the comparative information such as <char>, <qty> and <address> cannot be set when two serial operations are compared (in case that the value of direction attribute is “cmb”).

When “#location” is set in the value of spec attribute, the location information is compared. When the contents of <address> tags indicating the location information agree with each other, the character strings such as name attribute or uri attribute agree or the numerical values of east attribute, south attribute or height attribute agree. If some of these attributes are specified, all of them agree. When the size relation is compared, only one of east attribute, south attribute and height attribute can be set.

The case where anything is checked on the second floor of Koganei Factory is expressed as below.

```
<condition spec="address" type="EQ"><address name="Koganei factory" height="2"/></condition>
```

3.8. Expressing Order

Order expresses the various requests about production and requests item or resource and operation execution. Order is expressed with a <order> tag. Order can be divided into three types, item order, resource order and operation order according to the configuration.

Item order is the most popular order and requests only the fixed quantity of any item as a lot and sets ID of the item in item attribute.

In case of item order, the order duetime that is the latest completion date of operation to meet the request can be specified with <duetime>. And also the order duetime can be specified with an order release <release> tag, which is the earliest start date of operation to realize the order.

The following example shows the order for 150 of product X required by noon on April 20.

```
<order name="K003" item="product X">
<qty value="150">
<duetime><time value="2003-04-20T12:00:00"/></duetime>
</order>
```

The difference between item order, resource order and operation order is judged by the value set in item attribute in a <order> tag. Therefore the overlapping values cannot be set in item attribute,

resource attribute and operation attribute.

Resource order requests the fixed resource capacity as a task, not item as a task. Resource order specifies the ID of intended resource with resource attribute. The start time and end time of the period when the required resource capacity is needed are specified with a <release> tag and a <duetime> tag respectively.

The request that five labors come for help from April 11 to 13 is expressed as below.

```
<order name="K004" resource="labor">
<qty value="15" unit="person"/>
<release><time="2003-04-11T00:00:00"/></release>
<duetime><ime=" 2003-04-14T00:00:00"/></duetime>
</order>
```

When a time unit is set in multiplier attribute in a <resource> tag of the intended resource, the unit of resource order is the total value by time. For instance, if the required number and unit are 15 persons and "day" is specified in multiplier attribute, 15 persons/day is indicated.

In such a case, an order release tag <release> and an order duetime tag <duetime> are the earliest date and the latest date of task assignment respectively and so the term to provide the resource capacity is free within this period. In short, the above example means that it doesn' t matter whether 15 persons come for help on April 11 or 5 persons equally come each day.

The maximum value and the minimum value available to be set at each instant can be specified with a <tasksize> tag showing task size.

Finally, the operation order corresponds to schedule and directly requests operation execution. Operation order must specify the ID of the applicable operation with operation attribute in an <order> tag.

In case of operation order, the quantity to be set, the <qty> tag indicates the number of operation units. Size of lot or task can be increased or decreased by the number of units of this operation.

For example, in case that the number of product A to be produced is

defined as 2 in a <operation> tag as the operation information, if the number of units of operation order is 3, the lot size of product A is 6.

3.8.1. Order Classification

Order classification is expressed by status attribute. The values of status attribute are “fixed”, “forcast”, “unofficial”, “temporal”, “done” and “cancel”. The meaning of each value is as below.

fixed	Indicate the definite order.
forcast	Order that is issued based on forecasting in the company and isn’t associated to customer order yet.
unofficial	Unofficial order. Unofficial order shows the future definite order within an error of one acceptable range beforehand.
temporal	Temporary order. The corresponding order is what was used for creating the temporary plan in order to make the duetime estimation.
done	Used for showing the history of order executed in the past
cancel	Indicate the order that was once set but finally deleted and, not executed.

3.8.2. Setting Price

The price can be set in order. The price information is expressed with a <price> tag. <price> tag indicating the price information is used for item <item>, resource <resource>, operation <operation> and event <event> besides order.

The price set in an order tag <order> shows the just price for the order. On the other hand, the <price> tag set in <item> indicating item or <resource> indicating resource is the price per unit.

The case where the unit price of product A is 2,000 yen and three of product A are ordered is expressed as below.

```
<item name=“product A”><price value=“2000” unit=“yen”/></item>
<order name=“K001 item=“product A”>
<price value=“6000” unit=“yen”/>
<qty value=“3” unit=“piece”/>
</order>
```

In case of resource, the cost to use resource once and the cost per one hour can be set. When the cost is per one hour, the unit of unit time must be specified with a <base> tag.

The following shows that it costs 700 yen to use machine A for one hour.

```
<resource name="machine A"><price value="700" unit="yen"><base unit="#hour"/></price></resource>
```

The price information of operation indicates the cost to execute operation one time. When operation is the master information, the value of multiplied operation units is the actual price as the cost to operate one unit. When the costs are different according to operation time, the cost per hour can be taken by using a <base> tag.

The price information of event indicates the cost required for executing the event one time.

3.8.3. Expressing Customer and Supplier

Customers are enterprises, organizations and persons issuing an order. On the other hand, suppliers are enterprises, organizations and persons receiving an order. Customer and supplier are expressed with a <customer> tag and a <supplier> tag respectively.

The defined ID of customer or supplier is specified as the customer attribute or the supplier attribute in the <order> tag indicating order. Either of them can be specified in order.

PSLX recommends that the item introduced as a standard in each industry be used for the detailed order items used in the general e-commerce. Or individual items defined with the <spec> tag approved as the PSLX specification can be used.

General order information can be expressed as below.

```
<customer name="G001">
<spec name="TEL"><char value="012-345-6000"/></spec>
<spec name="FAX"><char value=" 012-345-7000"/></spec>
<spec name="person in charge"><char value="Yasuyuki Nishioka"/></spec>
</customer>
```

3.8.4. Pegging Information

A chain of order which one order produces another order and moreover the second order produces the third order can be expressed with a <pegging> tag as the pegging between orders. <pegging> tag indicating pegging is set to an order tag <order> and expresses the ID of the origin order to produce the order.

For example, if two orders (order B and order C) are issued to producing A to meet order A from a customer, order B and order C are pegged to order A as below.

```
<order name="order A"/>
<order name="order B"><pegging order="order A"/></order>
<order name="order C"><pegging order="order A"/></order>
```

The quantity can be set in pegging between orders. Specifying the numerical value in a pegging tag <pegging> with a <qty> tag expresses how many orders are pegged to the upper order specified with <pegging>. This function must be handled at implementation level 2 and over.

When order X produces 10 products for customer order A and 15 products for customer order B, the tag is as below. This function must be handled at implementation level 2 and over.

```
<order name="customer order A"/>
<order name="customer order B"/>
<order name="order X">
<pegging order="customer order A"/><qty value="10"/></pegging>
<pegging order="customer order B"/><qty value="15"/></pegging>
</order>
```

Pegging has a direction. The side nearer to customer is lower and the order nearer to the upper processing on the raw material side is upper. For example, the case where order X is seen from the viewpoint of order A is expressed as below. This function must be handled at implementation level 3 and over.

```
<order name="order A">
<pegging order="order X" type="upper"/><qty value="10"/>
</order>
```

Pegging information reaches the most original order by retracing the relation between orders. When the order is referred to with the

pegging <pegging> information and it indicates the end of order route where there is no order later, the final attribute must be ”true”. This function must be handled at implementation level 3 and over.

3.8.5. Direction Division

The division that is whether the order request is realized as early as possible or as late as possible can be specified in the order with the direction. Direction division is expressed by specifying direction attribute to an <order> tag.

“forward” indicating the case where the order is executed as early as possible and “backward” indicating the case where the order is executed as late as possible can be set in the direction attribute, the direction division.

Direction attribute can also be set in a <operation> tag indicating operation like order. When the value of direction attribute directly set to operation is different from the value set in the order information, the operation information must be selected. However the instance information must be selected, not the master information.

3.9. Expressing Operation

Operation is the basic unit to execue production and scheduling problem is expressed based on the operation. Operation has two types: the abstract information registered as a master beforehand and the operation corresponding to the schedule set in each time to each resource as the instance information as a result of making an actual schedule.

These two kinds of information are expressed with an <operation> tag. In case of the former abstract operation, the master attribute must be “true”. However when it is clear that all operations are the master information accorindg to interface type, it can be omitted to set the master attribute.

When producing the later instance information as an individual schedule based on the former abstract operation information, the ID of the former operation as a master can be specified with parent

attribute.

The order information producing the operation as the instance information can be expressed with order attribute.

The direction to assign the operation can be specified in operation with direction attribute like order. When the value of this attribute is ”forward”, it shows that the operation is executed at the early date as much as possible, and when the value is “backward”, the operation is executed at the latest time of duetime.

Type attribute is the division of operation. “make” in type attribute is for the ordinary production; “stock” is for storaging; “move” is for moving and “check” is for checking.

3.9.1. Operation Time

Operation time indicates the time from the start time of the operation to the end time. Operation time is expressed with a <duration> tag. If the operation is suspended during the period from start to end, the real operation time is different. In such a case, the net time can be specified by putting “true” in net attribute.

When defining the operation time as the master information, the information about how to calculate operation time can also be defined. When the operation time is fixed in spite of the lot size to be produced, type attribute in a duration tag is ”fixed”.

The following shows that the operation time is 120 minutes and fixed. The basic unit of operation time is second in this case and so 120 minutes are changed in terms of second.

```
<operation name="operation A"><duration value="7200" type="fixed"/></operation>
```

There are many cases where operation time is variable to the processing content in the usual operation. Thus when the processing time is different according to the amount of operation, fixed attribute is omitted. And so the time that the operation unit is multiplied by the time set with value attribute is the total operation time.

The following example shows that the amount of this operation is

three units and the unit time of operation is 30 minutes (180 seconds) and so 90 mimutes are required as the total amount.

```
<operation name="operation B" master="true">
<duration value="1800" />
</operation>
<operation name="B001" parent=" operation B">
<qty value="3"/>
<start><time value="2003-04-10T00:00:00"/></start>
<end><time value="2003-04-10T01:30:00"/></end>
</operation>
```

3.9.2. Expressing Event

The unit to execute any action is expressed as an event with a <event> tag. <start> tag indicating operation start and <end> indicating operation end are one kind of events.

<release> tag showing order release and <duetime> tag showing order duetime are especially given as a kind of events.

The event name cannot be set with name attribute in a start tag <start>, an end tag <end>, an order release tag <release> and an order duetime tag <duetime>. These events must be able to be identified with ID, operation name #start, operation name #end, order name #release or order name #duetime by checking with the operation name or the order name in which the event is set. Therefore the order names adding #release or #duetime at the end must not be permitted for producing order and the operation names adding #start or #end at the end must not be permitted for producing operation.

3.9.3. Producing and Consuming Item

The content to produce or consume an item can be specified in an <operation> tag indicating operation and an <event> tag indicating event. Producing item is expressed with a <produce> tag and consuming item is expressed with a <consume> tag. When the quantity of production or consumption isn't one unit, the quantity can be expressed with a <qty> tag.

The case where operation A uses 10 units of raw material Y and produces one unit of product X can be expressed as below.

```
<operation name="operation A" master="true">
<produce item="product X"/>
```

```
<consume item="raw material Y"><qty value="10"/></consume>
</operation>
```

When consuming or producing multiple kinds of items at the same time, nth attribute is set with changing the value. When some values are set in a <produce> tag and a <consume> tag by changing nth attribute, it means that each item is produced and consumed at the same time.

```
<operation name="operation B" master="true">
<produce item="product X" nth="1"/>
<produce item="odd material" nth="2"/>
<consume item="raw material Y" nth="1"/>
<consume item="parts Z" nth="2"/>
</operation>
```

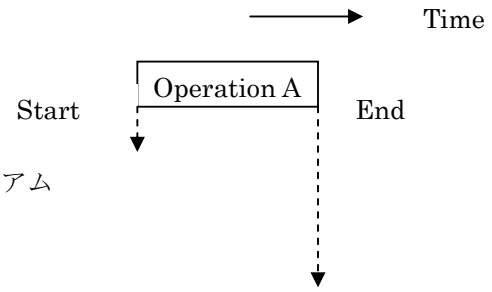
When selecting one produced itme out of some items, the same value is set in nth attribute. The information with the same nth attribute set in the same item tag <item> must be limited to one when the final schedule can be executed.

If operation is the instance information, lot attribute can be specified in production <produce> data and consumption <consume> data instead of item attribute. The ID specified with this lot attribute is an identifier of the lot that is actually produced or consumed by the operation.

For example, when operation A produces two units of product X, the state that an instance of operation A, A001 produces the lot of product X, X001 by the actual order is expressed as below.

```
<operation name="A001" parent="operation A">
<produce item="operation A" lot="X001"><qty value="2"/></produce>
</operation>
```

The stock value of items usually increases at the end of the operation as a result of production <produce> by operation. Only the specified quantity of items to be consumed by consumption <consume> decreases at the start point of operation.



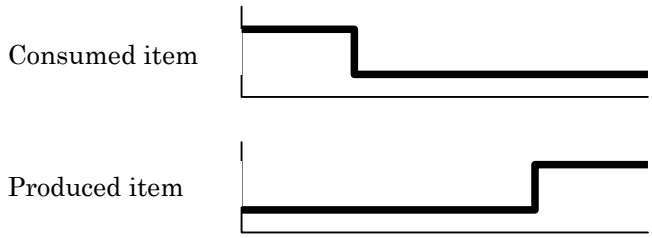


Figure　Production and Consumption by Operation（Discrete Type）

When type attribute is ”cont”, the amount of items continuously changes from start to end as the following figure shows.

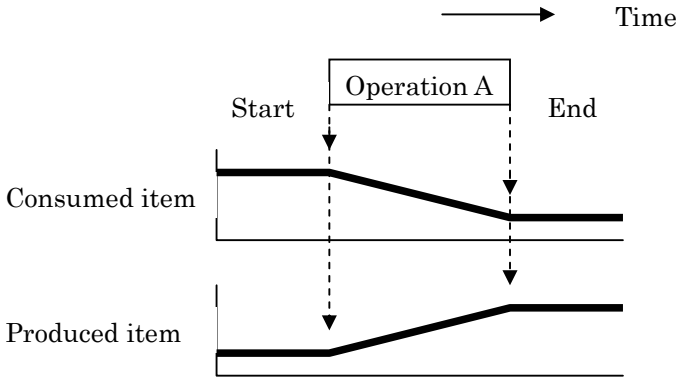


Figure　Production and Consumption by Operation (Continuous Type)

3.9.4. Assigning Resource

The information on the resource used to execute the operation can be specified in operation according to the assignment relation. The assignment of resource is expressed with a <assign> tag. It can be specified how much capacity of the resource is consumed by the operation. The load value for assignment can be specified using a <qty> tag indicating the quantity.

As the easy case, the information that operation A uses machine B is given as below.

```
<operation name="operation A">
<assign resource="machine B"/>
</operation>
```

The following shows that operation B needs three labors.

```
<operation name="operation B">
<assign resource="labor"><qty value="3" unit="person"/></assign>
</operation>
```

The load value required for assigning is the unit of work value that the set unit is multipled by time in a <resource> tag of the target resource when the time unit is set in multiplier attribute. This function must be handled at implementation 3.

When using some resources at the same time, changing nth attribute

sets the value. When selecting only one resource out of some resources, <assign> tag is set with the same value of nth attribute.

When either machine X or machine Y is used, it is specified as below. In the following case, nth="1" is the abbreviation value, so it can be omitted.

```
<operation name="operation C">
<assign resource="machine X" nth="1"/>
<assign resource="machine Y" nth="1"/>
</operation>
```

In case of assignment <assign> of resource, the continuous capacity consumption can be expressed by setting "cont" in type attribute like production <produce> or consumption <consume>. The value gradually increases from the operation start toward the operation end and from 0 to the specified load value. This must be handled at implementation level 3.

For example, the following describes the operation that the level gradually goes up for first one hour and after executing the steady operation for three hours, the level is gradually down for one hour and the operation ends.

```
<operation name="start operation">
   <start><time value="2002-09-01T00:00:00"/></start>
   <end><time value="2002-09-01T01:00:00"/></end>
   <assign type="cont" resource="X"><qty value="100"/></assign>
</operation>
<operation name="steady operation">
   <start><time value="2002-09-01T01:00:00"/></start>
   <end><time value=" 2002-09-01T06:00:00"/></end>
   <assign type="disc" resource="X"><qty value="100"/></assign>
</operation>
<operation name="end operation">
   <start><time value="2002-09-01T05:00:00"/></start>
   <end><time value=" 2002-09-01T06:00:00"/></end>
   <assign type="cont" resource="X"><qty value="-100"/></assign>
</operation>
```

3.9.5. Alternative Resource, Alternative Item

The consumed item, the produced item and the assigned resource may be selected out of some choices to one operation. Such the information about alternative item or alternative resource can be expressed with nth attribute.

And also these three selections may have the close relation with each other. In short, a selection result of the consumed item or the assigned resource may affect the selection of the produced item. These complicated selection relations can be expressed with nth attribute and sel attribute.

Nth attribute expresses what can be selected at the same time. Production, consumption, and assignment can be executed as much as the number of kinds of this value. In other words, when nth attributes are the same, finally 1 must be selected for an executable schedule. The value of nth can be set independently in <produce>, <consume> and <assign>.

In the following case, operation X produces item A and item B together but operation Y produces either item C or item D because both the values of nth attribute are one.

```
<operation name="operation X">
  <produce item="item A" nth="1"/>
  <produce item="item B" nth="2"/>
</operation>
<operation name="operation Y">
  <produce item="item C" nth="1"/>
  <produce item="item D" nth="1"/>
</operation>
```

The above nth attribute can originally be set in the produced item, the consumed item and the assigned resource but each selection may be related to other selections each other. Sel attribute is used for associating individual selections and for expressing the complicated relation of combining AND and OR.

This sel attribute can be set in a <produce> tag indicating a produced item, a <consume> tag indicating a consumed item and an <assign> tag indicating an assigned resource, but the value is only one in upper <operation>, the operation information. In short, if the value of sel attribute is selected for the assigned resource, the result is reflected to other selection elements.

The following shows that when operation X selects sel number “1”, item B1 is produced from item A1, and when operation X selects sel number “2”, item B2 is produced from item A2.

```
<operation name="opeation X">
<consume item="item A1" sel="1"/>
<consume item="item A2" sel="2"/>
<produce item="item B1" sel="1"/>
<produce item="item B2" sel="2"/>
</operation>
```

Nth attribute and sel attribute to indicate an alternative item and an alternative resource must be the serial integers from 1. Sel attribute must be handled at implementation level 3.

3.10. Precedence of Operation

3.10.1. Expressing Precedence

Precedence can be set in operation <operation> and event <event>. Firstly the precedence like the following types can be expressed to two operations <operation>.

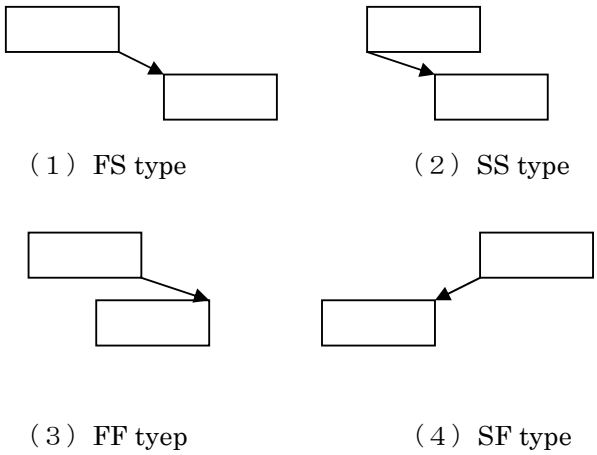


Figure Precedence Types

Precedence is expressed with a <predecessor> tag or a <successor> tag. When the party operation is a predessor of the operation to set the precedence, a <predecessor> tag is used, and when the party operation is a successor, a <successor> is used.

There are two ways to define the precedence between two operations. One is specifying a successor with a <successor> tag by a predecessor and the other is specifying a predecessor with a <predecessor> tag by a

successor.

When one definition is executed, the other definition must be able to be omitted. When both definitions are executed, there must be no contradictions.

When defining the precedence to three operations and more, there may not be a loop on the route tracing the predecessor or the successor.

The example that the successor of operation A is operation B is expressed as below. In the following example, the value of type attribute is omitted because it is ”FS”.

```
<operation name=“operation A”>
<successor operation=“operation B”/>
</operation>
```

This relation can be expressed from the viewpoint of operation B as below and these relations have the complete same meaning.

```
<operation name=“operation B”>
<predecessor operation=“operation A”/>
</operation>
```

When the fixed hours and more must be taken between the precedessor and the successor, <duration> is set in the tag of precedence. If <duration> is omitted, it means that the value is 0 and more. The following example shows that the interval between operation A and operation B must be 60 minutes and more.

```
<operation name=“operation A”>
<successor operation=“operation B”><duration value=“3600”/></successor>
</operation>
```

In the above example, if the time must be just 60 minutes, not 60 minutes and more, attribute type in <duration> is set as “fixed”. If the time must be 60 minutes and less, not 60 minutes and more, “short” is set in type attribute.

The precedence can be set in event <event> besides operation. The precedence can be set in the special events, start <start>, end <end>, order release <release> and order duetime <duetime>. The precedence to events must be handled at implementation level 2 and

over. The precedence to start, end, order release and order duetime must be handled at implementation level 3 and over.

3.10.2. Operation Switch

The setup time to two serial operations for any resource is changed according to the combination of serial operations. The information on such switch operation is specified with a <changeover> tag indicating switch relation.

<changeover> tag indicating the switch relation doesn't directly specify IDs of serial operations. Instead of that, the IDs are checked by showing the feature of former or later operation with a condition <condition> tag. If the result is true, the operation interval specified with a <duration> tag is applied. If there is no <condition> tag indicating condition, the setup time specified with a <duration> tag is always applied.

The way to set the condition to make the switch relation is divided into three types according to the value of direction attribute. Firstly, when the value of direction attribute is "pre", only the feature of just-before operation is a judging factor. Take-down is typical.

Secondaly, when the value of direction attribute is "suc", only the feature of just-after operation is a judging factor. This corresponds to setup. Thirdly, when the value of direction attribute is "cmb", the condition is judged by the combination of just-before operation and just-after operation.

The case where the condition is judged with only the feature of former operation or later operation is given. The following example shows that if the "setup" value of successor is "yes", 10 minutes is kept for setup.

```
<changeover>
<condition spec="setup" type="EQ" direction="pre">
<char value="yes"/></condition>
<duration value="600"/>
</changeover>
```

Thus, when only one of the former operation and the later operation is considered, <condition> tag has <qty> or <char> as a value in its own

right and the content is compared with the content of the target operation. If the result of comparing is true, the time set with <duration> is taken.

The following is the example that the features of both serial operations are considered. This is that when “solution kind” handled by the serial operations changes, it takes 60 minutes to setup. In such a case, there isn’t a <qty> tag or a <char> tag in a <condition> tag because the contents of specifications of predecessor and successor are compared.

```
<changeover>
  <condition spec="solution kind" type="NE" direction="cmb"/>
  <duration value="360"/>
</changeover>
```

When generally specifying the setup time for the predecessor and the successor, two <condition> tags with direction attributes, ”pre” and ”suc” are set to each. For example, the situation where it takes 20 minutes to switch type A to type B and it takes 30 minutes to switch type B to type C can be expressed as below.

```
<changeover>
<condition spec="Type" type="EQ" direction="pre"><char value="A"/></condition>
<condition spec="Type" type="EQ" direction="suc"><char value="B"/></condition>
<duration value="120"/>
</changeover>
<changeover>
<condition spec="Type" type="EQ" direction="pre"><char value="B"/></condition>
<condition spec="Type" type="EQ" direction="suc"><char value="C"/></condition>
<duration value="180"/>
</changeover>
```

The case where operation attribute is set in a <changeover> tag indicating switch means that the specified operation is required when the condition set with <condition> is fulfilled. This case corresponds to the case where setup must be operated separately.

3.10.3. Item Interval Period

The interval information of item is the concept that is very similar to the switch information of operation. This information specifies the interval from the time when any item is produced by any operation to the time when the item is consumed by another operation and it’s gone.

Such an interval data is expressed with a <interval> tag.

This constraint is used when the fixed time makes the quality of item poor such as perishable foods or when a product in the state of the specific work in process for a long time costs a great deal.

<changeover> tag indicating the switch data is defined in a <resource> tag indicating resource, while an <interval> tag indicating interval data is defined in an <item> tag indicating item.

The status where product A cannot be storaged in stock for 10 days and more can be expressed as below.

```
<item name="product A">
<interval><duration value="10" unit="day" type="short"/></interval>
</item>
```

The interval time can be set with considering one or both the attribute of producing and the attribute of consuming.

The following example shows that the 20-hour interval must be established only for special processing.

```
<item name="product B">
<interval direction="pre">
<condition spec="processing method " type="EQ"><char value="special processing"/></condition>
<duration value="20" unit="hour"/></interval>
</item>
```

When operation attribute is set in an <interval> tag indicating inverval, it means that the specified operation is required for fulfilling the condition set with <condition>. This corresponds to the case where a new operation is required for keeping the quality of the item.

3.11. Lot and Task

Lot is the item consumed or produced by the instance information of operation. Lot is expressed with a <lot> tag.

The ID for identifying is expressed with name attribute for lot. The quantity of lots produced or consumed is expressed with <qty>. Moreover the place where the lot is produced or consumed is expressed

with a <address> tag as occasion demands.

Lot can be divided into the lot produced by operation and the lot consumed by operation. This division is specified with type attribute. The relation between the produced lot and the consumed lot can be specified with a <tracking> tag as the tracking information.

When operation is the master information, the produced item and the consumed item are specified with a <produce> tag and a <consume> tag. The information of lot to be produced or consumed as a concrete substance can be set in the instance information of the operation.

When operation A produces two of product X, the concrete schedule A001 and lot, X001 indicating the concrete production of product X can be expressed as below.

```
<operation name="operation A" master="true">
<produce item="product X"><qty value="2" unit="piece"/></produce>
</operation>
<operation name="A001" parent="operation A">
<produce item="product X" lot="X001"/>
</operation>
<lot name="X001" type="produce">
<qty value="2" unit="piece"/></lot>
```

The lot corresponding to a <produce> tag, item production must set "produce" in the value of type attribute of lot. On the other hand, "consume" must be set in the lot produced to a <consume> tag, item consumption.

The quantity and the location information can be set in lot. The quantity shows the amount of products produced as a result. This value doesn't always agree with the value of <produce>. The location information of lot is the location information at the point of time when the lot is produced. And usually the location information of the facility used as the location to operate can be set in the location information of lot.

3.11.1. Lot Size Setting

The quantity of lots is basically calculated by adding the number of units of operation and the quantity set in a <produce> tag or a <consume> tag. Just the quantity specified by order may be set. The

lot size information must be handled at implementation level 2 and over.

However the value of lot size automatically calculated by an application can be controlled by beforehand setting the value in the item information <item>, the operation information <operation> and the order information <order> with a <lotsize> tag by the constraint of lot size. Lot size data set in the operation information is the lot size that nth attribute of the item produced by the operation is “1”.

When some lot size data compete with each other, firstly order data takes priority and the next is operation lot size and the last is the target item data. The instance information must always take priority over the master information. Moreover when unabling to judge the priority, the priority data specified with a <priority> tag is used.

When type attribute of a lot size <lotsize> is “fixed”, the set value is fixed. If the quantity is short, only the required schedules are added. When the values are “min” and “max”, the value set with “min” is for the lot size below this value and the value set with “max” is for the lot size over this value. When the value is “unit”, the multiple of the set value is set.

The following gives the case where the lot size of product A is set.

```
<item name=“product A”/>
<lotsize value=“30” type=“min”/>
<lotsize value=“200” type=“max”/>
<lotsize value=“10” type=“unit”/>
</item>
```

In the above example, the minimum lot size is 30 and the maximum lot size is 200 and the unit lot size is 10. If the order is 20, the lot size is 30 and if the order is 32, the lot size is 40. If the order is 230, two schedules are produced; lot size is 200 and lot size is 30.

3.11.2. Information on Task

Task is the unit that the operation uses resource in the actual schedule. Task indicates the state where the operation is assigned to resource and uses the capacity of resource actually. Task is expressed

with a <task> tag.

Task is the top-level information and can be identified by specifying the original ID name with name attribute. The quantity of tasks is expressed with <qty>. The tasks enough for the required capacity value are individually set to each resource assigned with operation. The place where the task is produced, that is to say the location of target resource can be set in each task with <address>.

Task may indicate the operation load at each instant and may indicate the amount of work with time axis (amout of operation). When the time unit is set in multiplier attribute of <task> tag indicating task, the value means the amount of work.

When operation A uses machine X, “X001” indicating the concrete usage of machine X can be expressed as below.

```
<operation name=“operation A”>
<assign resource=“machine X”><qty value=“3”/></assign>
</operation>
<task name=“X001” resource=“machine X”><qty vaue=“3”/></task>
```

<tasksize> tag indicating the constraint of task size can also be specified to a resource tag <resource>, an operation tag <operation>, and an order tag <order> like lot size. Even if the capacity of resource reserves power, the capacity of resource is used following the made rule by specifying task size.

If the number of part-timers is 10 and the maximum value of task size is 4, the schedule of work for 10 persons is 4 persons on the first day, 4 persons on the second day and 2 persons on the third day.

```
<resource name=“part-timer” multiplier=“day”>
<tasksize value=“4” type=“max”/></resource>
<operation name=“operation C”><assign resource=“part-timer”/></operation>
<task name=“operation C-001” resource=“part-timer”><qty value=“4”/></task>
<task name=“operation C-002” resource=“part-timer”><qty value=“4”/></task>
<task name=“operation C-003” resource=“part-timer”><qty value=“2”/></task>
```

<tasksize> tag set in operation is the task size information of resource with nth attribute “1” among the resources used by the operation. When the task size set in operation competes with the task size information set in resource and order, the order of priority are order,

operation and resource. However the instance information always takes priority over the master information.

3.11.3. Lot Tracking and Task Tracking

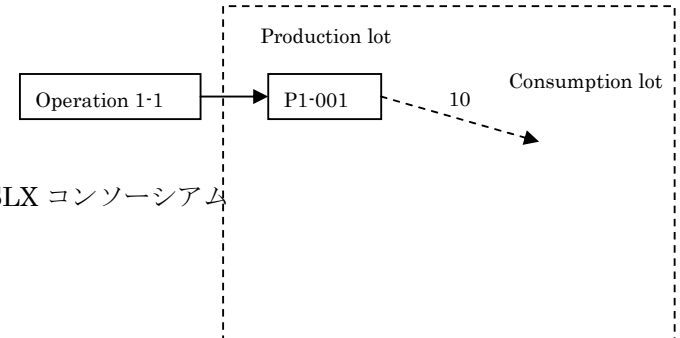
There are production lot and consumption lot in lot <lot>. When type attribute of a <lot> tag is “produce”, it is a production lot. When type attribute is “consume”, it is a consumption lot. Production lot is individually produced for production <produce> of item and consumption lot is individually produced for item consumption <consume> of item.

Lot tracking is available by associating a production lot with a consumption lot in the same item. Lot tracking data is expressed with a lot tracking tag <tracking>. This is the pointer information from the production lot to the consumption lot, or from the consumption lot to the production lot. Each correspondence can be set adding the quantity relation using a <qty> tag.

For example, if the former processing, operation 1 is executed three times and the later processing, operation 2 is executed two times in middle product A, the following lot tracking data can be defined.

```
<lot name=“P1-001”><qty value=“10”/></lot>
<lot name=“P1-002”><qty value=“10”/></lot>
<lot name=“P1-003”><qty value=“10”/></lot>
<lot name=“P2-001” type=“consume”><qty value=“12”/>
<tracking lot=“P1-001”><qty value=“10”/></tracking>
<tracking lot=“P1-002”><qty value=“2”/></tracking></lot>
<lot name=“P2-002” type=“consume”><qty value=“12”/>
<tracking lot=“P1-002”><qty value=“8”/></tracking>
<tracking lot=“P1-003”><qty value=“4”/></tracking></lot>
```

This shows the relation between lots as below. In the above example, the production lot is specified from the side of consumption lot, however conversely the consumption lot can also be specified from the side of production lot. The production lot cannot be associated to a production lot, and the consumption lot cannot be associated to a consumption lot by <tracking>.



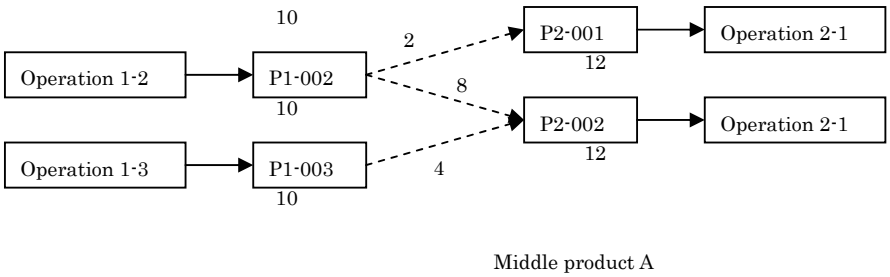


Figure　Lot Tracking

Meanwhile the relation between tasks can be expressed with a <tracking> tag, too. The relation between tasks is like the case where operation A-1 has 100 hours of operation value as a task and this task is divided into five in each 20 hours and each task is executed in parallel by five machines. Such a case can be described as below.

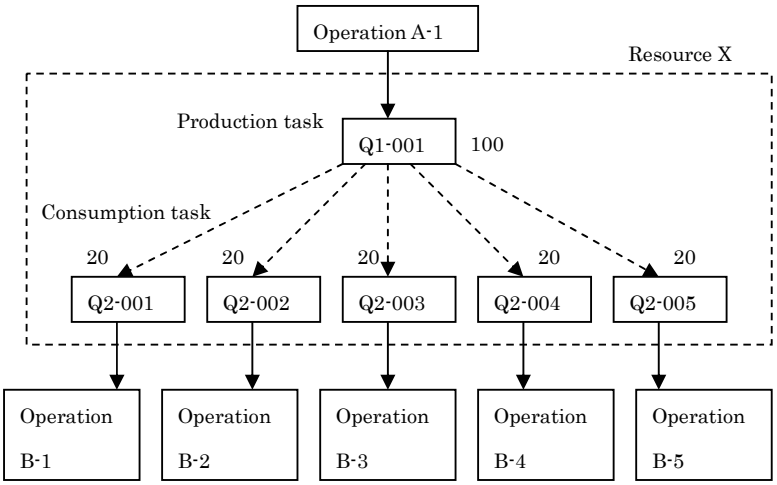


Figure　Task Tracking

3.12. Inquiring Data

3.12.1. Available Range of Value

The range of data must be specified for query when inquiring the data. In the PSLX tag structure, the available range of the values of <qty>, <price>, <priority> and <duration> indicating the numerical information or <char> indicating the character information can be specified. The earliest date and the latest date can be set in <time> indicating the date and time. The range specification must be handled at implementation level 2 and over.

On the numerical information, the value set with value attribute in the above tag is the defined value and the range can be specified with <max> and <min> tags besides the above tags. For example, when the range from 1,000 yen to 2,000 yen is specified, the tag is expressed as below.

```
<price>
<min value="1000" unit="yen"/>
<max value="2000" unit="yen"/>
</price>
```

When the numerical value isn't included for comparing, exclusive attribute can be used. When the value is under 2,000 yen in the

above example, the tag is set using exclusive attribute as below.

```
<price>
<min value="1000" unit="yen"/>
<max value="2000" unit="yen" exclusive="true"/>
</price>
```

In case of the character information, the selectable proposed value can be specified with a <enumerate> tag. The following example expresses the assembly consisting of two character strings, "yes" and "no".

```
<char>
<enumerate value="yes"/>
<enumerate value="no"/>
<char>
```

<earliest> tag indicating the earliest date and <latest> tag indicating the latest date can be used for specifying the available range of date. The following example shows the period from 8 o'clock to 19 o'clock on 2003/04/10.

```
<time>
<earliest value="2003-04-10T08:00:00"/>
<latest value="2003-04-10T19:00:00"/>
</time>
```

3.12.2. Setting The Way of Inquiring

When inquiring the content of data by a request message, the content to be inquired must be specified concretely. The data of basic elements, item, resource, lot, task, operation, event, customer, supplier and order can be inquired by PSLX. The content of the numerical expression information and the parameter information registered as planning elements can also be inquired.

When inquiring data, the tag of element to be inquired is specified and the ID of the data to be inquired is specified in the value of name attribute or character string #query" is set. When specifying ID, the content of the applicable data is returned. When specifying "#query", all the data fitted for the retrieval condition are returned.

When restricting the intended elements, the retrieval condition is specified. In order to set the retrieval condition, attribute or

subelement is set in the element tag with name ”#query”. All the set contents are recognized as the constraint with the relation of AND and the retrieved targets are restricted.

If looking for the product of which price (unit price) is 1,000 yen and below, the tag is set as below.

```
<item name="#query">
<price><max value="1000"/></price>
</item>
```

The tags of the maximum value <max> and the minimum value <min> can be set in <qty>, <price>, <priority> and <duration> to restrict the inquired data. The tags of the earliest date <earliest> and the latest date <latest> can be set in the date and time <time>.

When the value is a character string, not a numerical value or date, some proposed values can be specified with a <enumerate> tag. In the following example, colors, ”red”, ”yellow” and ”blue” are selected.

```
<item name="#query">
<spec name="color"><char>
<enumerate value="red"/>
<enumerate value="yellow"/>
<enumerate value="blue"/>
</char></spec>
</item>
```

All the target data are limited according to the relation of AND in the attribute information and the subelement information set in the basic element. When the same kind of basic element is specified multiple times, they have the relation of OR.

<query> tag is used for specifying the data item to be actually inquired for the retrieved target. The item specified with select attribute in this tag is selected as the item of inquiry result. The tag name of subelement of the element can be set in select attribute.

If a <query> tag isn’t specified, only ID of the applicable data is returned. However <query> cannot be set for inquiring <parameter> indicating parameter and <expression> indicating constraint expression. In these cases, all the data are returned.

In the following case, the information about the start date and the end date of operation "OP001" and the assigned resource are returned.

```
<operation name="OP001">
<query select="#start"/>
<query select="#end"/>
<query select="#assign"/>
</operation>
```

The value of name attribute is always set in the attribute data of basic element as the data item returned to inquiry. Other attribute values are all returned when the value is select="#attribute" by a query tag. When "#all" is specified in the value of select attribute, all the information of the element are returned.

The content set with a <spec> tag as the specification information of target element can be inquired by directely specifying the value of name attribute in a <spec> tag in select attribute. When the value is select="#spec", all the specification data are returned.

All the data need not always be returned to an inquiry request. The application can constrain the content of information returned to the inquiry according to the party. If the inquired data item cannot be responded by the judgement of application side, an alarm message or an error message must be returned.

3.12.3. Inquiring The Content of Subelement

All the data set within the range of implementation level are returned as the content of subelement of the inquired basic element. However only the attribute and the <time> tag information in each of subelements of operation tag <operation> --- <ev>, <start> and <end> --- are returned.

Therefore the response to the former inquiry is as below.

```
<operation name="OP001">
<start><time value="2003-05-02T12:00:00"/></start>
<end><time value="2003-05-02T14:00:00"/></end>
<assign resource="machine A"><qty value="1"/></assign>
</operation>
```

When some same subelements exist in the inquired basic element, all the data of subelements are returned as a rule. However the

intended period can be limited with start attribute and end attribute in a <query> tag or a <profile> tag for the time series information such as specification information, location information, progress informaton, load information, stock information and calendar information. If start attribute and end attribute are omitted in each case, it is regarded that the dates equivalent to #init and #final are set in each attribute.

When the intended period information is set with start attribute and end attribute, the server receiving a message must select only the information included in the intended period out of some time series data and return the result. However one of data with the date value, #init can be included in the inquiry result as the data out of the specified period.

When inquiring the content of expression information <expression> and parameter information <parameter> as a planning element, subelement cannot be specified with a <query> tag. Therefore all the related data items are returned for these element data.

When other basic elements are referred to as a result of inquiry, those basic element data can be added to the top level. And when referring to the top-level data such as <unit>, <scale> and <shift>, those elements can be added, too. The description of referred element data is only ID in case of basic element data. The data definitions other than the above data definition can be shown. When referring to other elements from those attached data such as unit attribute in a <scale> tag, the element can be included, too.

3.12.4. Inquiring Hierarchical Structure

The result of repeatedly tracing the hierarchical structure information such as Manufacturing BOM information can be inquired. The depth parameter is given with depth attribute and other parts are set like the usual inquiry to order to inquire a chain of hierarchical structure. The chain inquiry function to the hierarchical structure must be handled at implementation level 3.

For example, an item connectes to another item through operation in Manufacturing BOM and this operation is repeated. In lot tracking,

lots are associated continuously through operation. Moreover orders are associated continuously through the pegging information.

The values that can be set in select attribute as the inquiry of hierarchical structure are “produce” or “consume” indicating Manufacturing BOM, “tracking” indicating lot tracking, “pegging” indicating pegging, “predecessor” or “successor” indicating precedence and “partof” indicating parent-child relation.

When the value of depth attribute indicating depth is 1, the primary basic element, which is retrieved like the ordinary inquiry, is returned. When the value is larger than it, the basic element is retrieved in as deep as the number of the value. When tracing the depth direction of the final network structure, the maximum number of times of basic element appearing is the value of depth attribute.

Minus value can be set in the value of depth attribute indicating depth, only when the value of select attribute is ”partof”. In such a case, the retrieval direction is reverse and the basic element that the element is parent (in short, child) is inquired.

The following figure shows the relation with the inquired element. The part indicated with a vertical arrow is the target inquired directly and the element name on the right is specified in select attribute. The number in parentheses is the part corresponding to the value of depth attribute. And the element that this number is below the value of depth attribute is included in the inquiry result. The end of figure is connected with the opposite end.

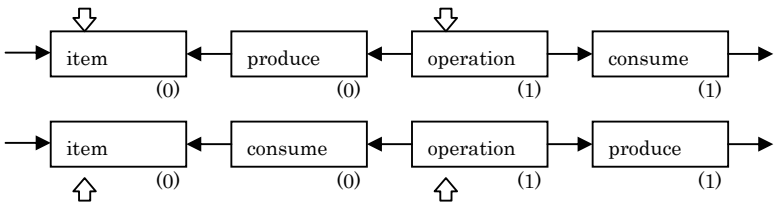


Figure Inquiring BOM Information

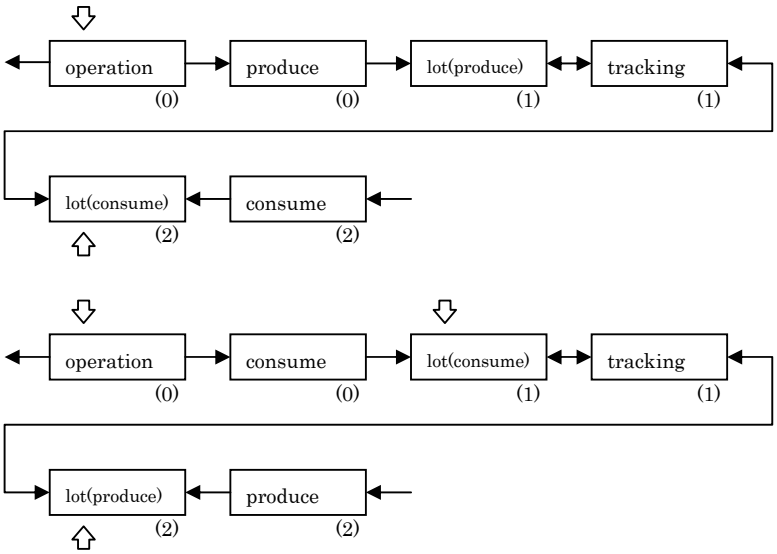


Figure　Inquiring Lot Tracking Information

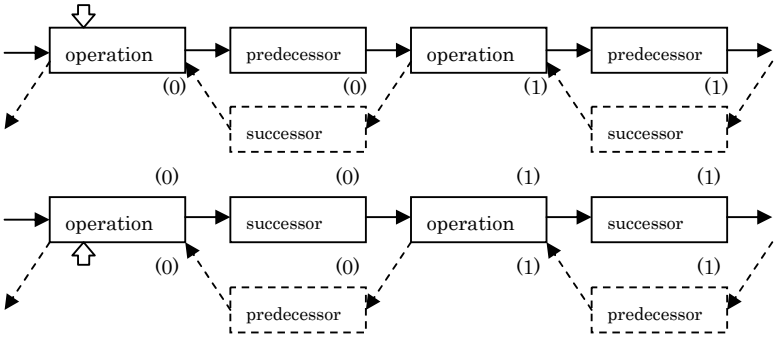


Figure　Inquiring Precedence

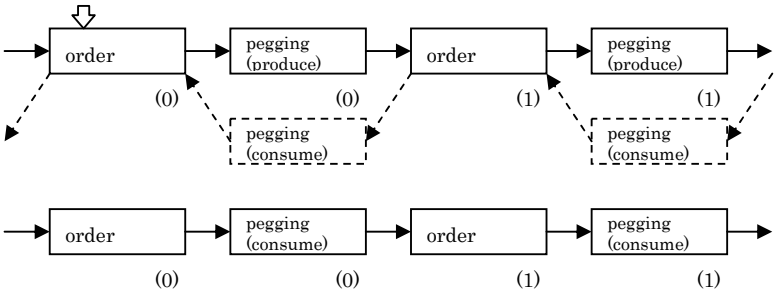


Figure　Inquiring Pegging

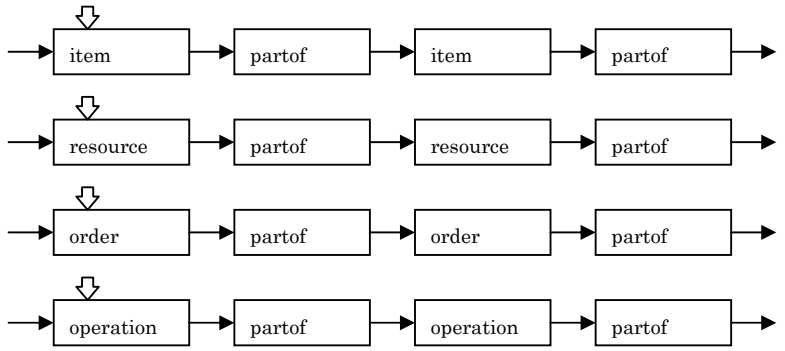


Figure　Inquiring Inclusion (Parent-child Relation)（depth＞0）

In the above figure, the elements inquired directly are retrieved in the left direction and sometimes the retrieval direction is opposite to the direction of arrow.　This means that the next element isn't retrieved directly from the inquired element and conversely the element is retrieved from the next element.

The element indicated with a broken line means that the element can get to the basic element on the next level, not through the right-side lement.　When there is no path indicated with a solid line, the path indicated with a broken line must be confirmed as well.

3.13.　Setting Subject

The results of planning and scheduling need to be seen multilaterally from various viewpoints.　Subject is the information that calculates individual data in various ways to respond to this request and makes the value available to be inquired.　Subject can be set in basic elements of customer, supplier, item, resource, lot, task, event,

operation and order and a <parameter> tag indicating parameter.

For setting subject, the value of name attribute is “#query” and the value of calculation attribute is any of the below attribute values when defining each basic element. The calculated data are only data that are adapted for the limitation condition specified in setting the basic element by the same way as inquiring. This function must be handled at implementation level 3.

Attribute value	Explanation
cnt	Count the number of existing elements
sum	Calculate the total value of intended data
ave	Calculate the average value of intended data
min	Calculate the minimum value of intended data
max	Calculate the maximum value of intended data
earliest	Calculate the earliest date of intended data
latest	Calculate the latest date of intended data

3.13.1. Restricting Calculated Item

Which data item is calculated to individual element is specified with select attribute in a <query> tag. Tag name of subelement of basic element can be specified with this select attribute. However a specification name of the specification information that has ”address”, ”description”, ”display” and a character string as an element cannot be specified. Moreover “lotsize”, “tasksize”, “condition”, “action”, “changeover” and “interval” related with the constraint condition cannot be specified as well.

When start attribute and end attribute are specified in <query> tag indicating subject, the calculated data is limited in the period. In other cases, when start attribute and end attribute are set in <profile> indicating the profile information, the data is limited in the period.

When calculating the data, the denominator specified with <base> and the unit specified with unit attribute must be considered.

The values that can be set as a value of select attribute for calculation

are given as below. In such a case, whether the numeric information can be calculated or not is described with the tag structure.

Basic element	Select attribute	Corresponding numerical information	Aggre-gation
Common	Name of spec	<spec name="aaa"><qty value="xxx"/> </spec>	○
Common	priority	<priority value="xxx"/>	
item	price	<price vale="xxx"/>	
item	capacity	<capacity><qty value="xxx"/> </ capacity>	○
item	stock	<stock><qty value="xxx"/></ stock>	○
item	(produce)	<produce><qty value="xxx"/></produce>	○
item	(consume)	<consume><qty value="xxx"/></consume>	○
resource	price	<price vale="xxx"/>	
resource	capacity	<capacity><qty value="xxx"/> </ capacity>	○
resource	load	<load><qty value="xxx"/></ load>	○
resource	(assign)	<assign><qty value="xxx"/></assign>	○
lot	qty	<qty vale="xxx"/>	
lot	(produce)	<produce><qty value="xxx"/></produce>	○
lot	(consume)	<consume><qty value="xxx"/></consume>	○
task	qty	<qty vale="xxx"/>	
task	(assign)	<assign><qty value="xxx"/></assign>	○
event	price	<price vale="xxx"/>	
event	time	<time vale="xxx"/>	
event	predecessor	<predecessor><duration value="xxx"/> <predecessor>	○
event	successor	<successor><duration value="xxx"/> <successor>	○
event	produce	<produce><qty value="xxx"/></produce>	○

event	consume	<consume><qty value="xxx"/></consume>	○
operation	price	<price vale="xxx"/>	
operation	start	<start><time vale="xxx"/></ start>	
operation	end	<end><time vale="xxx"/></ end>	
operation	event	<event><time vale="xxx"/></ event>	○
operation	qty	<qty vale="xxx"/>	
operation	duration	<duration value="xxx"/>	
operation	progress	<progress><qty value="xxx"/></progress>	○
operation	predecessor	<predecessor><duration value="xxx"/> <predecessor>	○
operation	successor	<successor><duration value="xxx"/> <successor>	○
operation	assign	<assign><qty value="xxx"/></assign>	○
operation	produce	<produce><qty value="xxx"/></produce>	○
operation	consume	<consume><qty value="xxx"/></consume>	○
order	price	<price vale="xxx"/>	
order	qty	<qty vale="xxx"/>	
order	release	<release><time vale="xxx"/></ release>	
order	duetime	<duetime><time vale="xxx"/></ duetime>	
order	progress	<progress><qty value="xxx"/></progress>	○

The following expresses that the quantity of product X produced by machine A for 5 days from April 7 to 11 is inquired.

```
<operation name="#calculation" calculation="sum">
<produce lot="product X"/>
<assign resource name="machine A"/>
<query select="produce" start="2003-04-7T00:00:00"
end="2003-04-12T00:00:00"/>
</operation>
</aggregation>
```

The calculated value is specified as the specification information in a <spec> tag to respond to the inquiry. The content specified with select attribute adding "#" is set in the specification name. When specifying the name with the original specification, the name is set as it is.

When the item to be calculated is specified with select attribute, the element put in parentheses in the above table can be specified even if the element is a subelement that the actual target element doesn't have. This shows the content that can be calculated by tracing the relation between basic elements conversely even if the inquired element doesn't have the subelement. In such a case, the information with making a tag name a specification name is returned as a result of calculation. For example, the following case is given.

```
<item name="product A">
<spec name="#produce"><qty value="123000"/></spec>
</item>
```

3.13.2. Partial Aggregation

When aggregating the item data of capacity, progress, stock and load indicating the time series information, the period must be calculated based on the fixed rule beforehand because there are some data in every date and time in the individual calculated element. And also the partial aggregation is needed because there may be some data in one element: produce, consume, assign or event.

Calcuation attribute is set in the <query> tag set in the target element in order to aggregate each data partially. The concrete values that can be set here are start（head value or beginning period value）, end（end value or end period value）, sum（total）, ave（average）, max（maximum value）, min（minimum value）and cnt（quantity）. The abbreviation value of this subject is "end". In case of the time series information, the latest data value is applicable and in other cases, the end data value is applicable.

3.13.3. How to Aggregate Discretely

The period set with a <query> tag or a <profile> tag can be calculated by dividing it off with the fixed bucket for calculating each element

data besides the way to calculate the whole period. This is called discrete aggregation. For executing discrete aggregation, scale attribute is set in the <query> tag where the subject is specified and the name of discrete scale is specified as a value.

Thus the content aggregated in each period of scale is set in the content to be returned as an aggregation result. For example, the following expresses the case where the amount of product X produced by machine A for five days from April 7 to 11 is aggregated in every day.

```
<scale name="daily" unit="#day" type="disc"/>
<operation name="#query" calculation="sum">
<produce lot="product X"/>
<assign resource name="machine A"/>
<query select="produce" scale="daily" start="2003-04-7T00:00:00"
end="2003-04-12T00:00:00"/>
</operation>
</aggregation>
The calcuation results of this case are as below.
<operation name="#query" calculation="sum">
<spec name="#produce"><time count="0"/><qty value="200"/></spec>
<spec name="#produce"><time count="1"/><qty value="320"/></spec>
<spec name="#produce"><time count="2"/><qty value="140"/></spec>
<spec name="#produce"><time count="3"/><qty value="230"/></spec>
</operation>
```

When origin attribute is set in a <scale> tag indicating scale for discrete aggregation, start attribute set in a <query> tag is ignored.

3.13.4. Calculating by Parameter

The value of one parameter data can be calculated by aggregating. In such a case, only one of the aggregated basic elements is set in a <parameter> tag indicating parameter and the subject is specified in the element.

The parameter indicating the average quantity of stocks in May is as below.

```
<parameter name="May average stock value">
<item name="#query" calculation="ave">
<query select="stock" calculation="ave" start="2003-05-01T00:00:00"
end="2003-06-01T00:00:00"/>
</item></parameter>
```

The result of aggregation based on the content set in a <parameter>

tag by parameter definition must be set in value attribute of the parameter tag by processing such as plan execution on the application.

When the target data executes discrete aggregation for defining parameter, the content is accessed by specifying index attribute in a <parameter> tag. The values of index are integers from 0. The head of period is 0.

For example, when the average stock value in May is scattered in each week bucket, the different value is expressed by changing the value of index attribute in every period as below.

```
<parameter name="May average stock value" index="0" value="340/>
<parameter name="May average stock value" index="1" value="520"/>
<parameter name="May average stock value" index="2" value="490"/>
<parameter name="May average stock value" index="3" value="330"/>
<parameter name="May average stock value" index="4" value="420"/>
```

3.14. Expressing and Optimizing Expression Information

When the parameter to be decided for planning is the numeric information, constraint and evaluation function can be defined. <expression> tag is used for defining the numeric information. This function must be handled at implementation level 3.

3.14.1. Expressing Numerical Expression

<expression> tag indicating the numerical expression information can express constraint and evaluation. And a part of those expressions can be defined as another numerical expression information.

Item data, which are the constituents of numerical expression information, are expressed with a <op> tag. Each item data has the appearance order in the expression and "+", "-", "*", "/" as algebraic signs. The appearance order is expressed with no attribute and the algebraic sign is expressed with operator attibute.

For example, the formula, (A＋B)＊C is expressed as below.

```
<expression name="e01">
<op parameter="A" no="1"/>
<op parameter="B" no="2" operator="+"/>
<op parameter="C" no="3" operator="*"/>
</expression>
```

The reference destination of each item is one of parameter information, another expression information and constant. The reference destinations are set with parameter attribute, expression attribute and value attribute respectively. When referring to another numerical information, the reference relation may not be a loop.

For example, the expression, A＋（２＊B）is expressed as below.

```
<expression name="e02">
<op parameter="A" no="1"/>
<op expression="e03" no="2" operator="+"/>
</expression>
<expression name="e03">
<op value="2" no="1"/>
<op parameter="B" no="2" operator="*"/>
</expression>
```

3.14.2. Setting about Optimization

When making the decision such as optimization, the models indicating the problem can be expressed with a <expression> tag as evaluation or constraint. One model consists of some constraints and evaluations. This function must be handled at implementation level 3.

Attribute type tells evaluation from constraint. The value of type attribute is "MAX" or "MIN" in evaluation. And the value is "EQ", "NE", "GT", "LT", "GE", or "LE" in constraint.

The value on the right side indicating constraint is specified with rhd attribute. The value on the left side is set in value attribute. When constraint is illegal as comparing the right side and the left side, the difference information of violation attribute is set with a plus real number. Therefore when the value of violation attribute is plus, it shows that constraint is illegal.

The following mathematical model can be described.

```
max.    -2x+y
s.t.    x<=4
        y<=3
```

The evaluation and the constraint of the above model are expressed with three formulas, E1, E2, and E3 as following.

```
<expression name="E1" type="MAX">
```

```
<op no="1" operator="+" value="-2"/>
<op no="2" operator="*" parameter="X"/>
<op no="3" operator="+" parameter="Y"/>
</expression>
<expression name="E2" type="LE" rhd="4">
<op no="1" parameter="X"/>
</expression>
<expression name="E3" type="LE" rhd="3">
<op no="1" parameter="Y"/>
</expression>
```

4. PSLX Messaging Specification (Part 2)

4.1. PSLX Interface framework

The information about the production planning and scheduling expressed with XML is exchanged between business processes of various APS agents. This chapter defines the interfaces of APS agents. The defined interfaces correspond to the interfaces of Planning agent, Scheduling agent and Federation management agent in "APS Agent Models" in Part 2 of PSLX Engineering Specification. (setEmergency and getEmergency aren't used.)

The following table shows the interface names defined in this specification and the simple explanations. The application based on XML Standard Specification Part 2 in PSLX must implement all functions of these interfaces or return an error message telling that the non-implemented interface is called.

Interface name	Explanation
initPlan	Initialize plan
makePlan	Make plan
initSchedule	Initialize scheduler
makeSchedule	Make scheduling
setCalculation	Set calculation
getCalculation	Inquire calculation
setPlan	Set plan
getPlan	Inquire plan
setSchedule	Set schedule
getSchedule	Inquire schedule
setParty	Register the customer and supplier information
getparty	Inquire the customer and supplier information
setProduct	Register the product (master) information
getProduct	Inquire the product (master) information

setProcess	Register the process (master) information
getProcess	Register the process (master) information
setOrder	Register the order information
getOrder	Inquire the order information
setEstimation	Set the order estimation
getEstimation	Request the order estimantion
setProgress	Set the progress information
getProgress	Inquire the progress information
setStock	Set the stock information
getStock	Inquire the stock information
setLoad	Set the load information
getLoad	Inquire the load information
setCapacity	Set the capacity information
getCapacity	Inquire the capacity information
setCalendar	Set the calendar information
getCalendar	Inquire the calendar information
setLot	Set the lot information
getLot	Inquire the lot information
setTask	Set the task information
getTask	Inquire the task information

An interface name consists of a prefix such as set, get, init, make, and a following character string. Prefixes, set and get, indicate the action realized by the interface and the following character string indicates the target for action.

A request message, a response message, a receipt message and an exception message are defined to each of these interfaces. The request message to notify a server of the request from a client has the form as below.

```
<pslx type="request" id="message ID" action="interface name">
・・・
request
・・・
</pslx>
```

The server executes the required processing to this request message and returns the result to the client with a response message. The form of response message is as below.

```
<pslx type="request" id="message ID" ref="request message ID" action="interface
  name">
・・・
response
・・・
</pslx>
```

The server may send an exception message besides a response message. The server and the client must return a receipt message when a receipt confirmation request is set in a request message or a response message.

The following section describes the detailed explanation for exchanging messages between the server and the client.

4.2. Request Message

The interface with a prefix, set, is to send the business information to the party agent. The interface with a prefix, get, is to receive the information about business from the agent. The request message requests service to these interfaces.

The business information sent with a request message is the information on basic elements, client, supplier, item, resource, lot, task, operation and order and the parameter information on plan and the calculation expression information.

All these basic elements and the elements related with the plan making a request message must be set on the level just under <pslx> tag. Using ID must enable all the individual data to be identified in this basic element information and the information on plan.

The type attribute value in a <pslx> tag must be set at “request” to show a request message. The optional character string that is unique in the requested application must be set in id attribute as the code to identify a message. The interface name must be set in action attribute.

A receipt request can be added to a request message to confirm that the requested application receives the message. Setting the receipt attribute in a <pslx> tag at “True” can change a request message to the request message with receipt confirmation. However, in case of synchronous communication, the receipt confirmation cannot be requested.

4.2.1. Set/Revise Element data

The interface with a prefix, set, requests a server to add, correct or delete element data. When adding the element data, “create” must be set in the action attribute in a tag of the added basic element or plan element. When correcting or deleting the data, “resive” and “delete” must be set in the action attributes respectively.

The server adding, correcting and deleting the element data must return only the ID of the element data that can be processed normally as a response message after processing. If the requested processing fails in a part of element data, the error information including the ID of the failed element data is set in an <error> tag and must be returned to a requester.

When all requests fail, an exception message must be returned. Even if a part of the requests succeeds, a response message must be returned.

When a request message is the addition of element data and the added object exists on the server receiving a message, the warning information must be set in an <error> tag and must be returned as a response message as a result of processing. In such a case, all the existing data are deleted once and then the newly set data is added.

When a request message is the correction of element data and no corrected object exists on the server, the data must be added newly. After that, the warning information must be set in an <error> tag in a response message and must be returned.

When a request message is the deletion of element data and no deleted object exists on the server, the error information must be set with an <error> tag and must be returned to a requester. If all requests fail,

an exception message must be returned. Even if a part of the requests succeeds, a response message must be returned.

When the server has no authority to add, correct and delete the element data, the error information is set with an <error> tag and then an exception message must be returned.

One request message can execute only one instruction of adding, correcting and deleting.

When correcting or deleting a basic element, the element data to be corrected or deleted cannot be corrected or deleted in a lump with name="#query".

4.2.2. Revise Sub-Element data

All the subelements and attributes of the target element data existing on the server are overwritten for correcting the element data. However, the subelement and the attribute that aren't set as the corrected contents aren't overwritten and so the former information remains as it is. If the value isn't set in the attribute and also the omitted value is defined, the omitted value is set newly.

Some basic elements of the target element data can have multiple data as subelements according to the basic element type. In such a case, whether the newly specified subelement contents replaces the existing subelement contents or whether the contents are newly added with leaving the existing subelemnt contents as they are depends on circumstances.

Firstly, when a subelement is the time series information and has some data in every date and time, the earliest date and later in the data set for correcting replace the former data newly. When the set date and the existing date are the same, the same time is left in case that the relative attribute is true and is replaced with the new data in other cases.

For example, the stock data on May 1, 3 and 5 exist as below.

```
<item name="product A">
<stock><time value="2003-05-01T00:00:00/><qty value="200"/></stock>
<stock><time value="2003-05-03T00:00:00/><qty value="260"/></stock>
<stock><time value="2003-05-05T00:00:00/><qty value="140"/></stock>
```

```
</item>
```

The stock information on May 2 is newly set under this state as below.

```
<pslx type="request" id="123" action="setStock"/>
<item name="product  A" action="revise">
<stock><time value="2003-05-02T00:00:00"/><qty value="290"/></stock>
</item>
</pslx>
```

In the case, the stock data on May 2 and later set newly are deleted and so the data is together with the data on May 1 as below.

```
<item name="product  A">
<stock><time value="2003-05-01T00:00:00/><qty value="200"/></stock>
<stock><time value="2003-05-02T00:00:00"/><qty value="290"/></stock>
</item>
```

When deleting all the time series information, <time ref="#init"/> is set and a <qty> tag is omitted.

The element has some sublements except the time series information in case that the name attribute in a <spec> tag indicating specification is different and in case that the name attribute in an <event> tag indicating event is different. When correcting these data, new contents replace the existing contents by identifying a name attribute value.

<produce>, <consume> and <assign> in the elements with some subelements give sense to the subelements with nth attribute and sel attribute. In such a case, all the usual data that the values of the newly set nth attribute and sel attribute are the same are deleted and overwritten.

<predecessor>, <successor>, <partof>, <pegging>, <tracking>, <interval> and <changeover> indicating the relation with other basic elements have some subelements besides the above elements. In these data, the corrected subelements are distinguished with the combination of the target element data ID and another element ID set in a tag and then the subelement contents are corrected. But when the element has still some applicable subelements, the contents are corrected after deleting all the applicable subelements.

If deleting just the subelement, not correcting the subelement contents, other related element IDs are set at “#query” in the above <produce>, <consume>, <assign>, <predecessor>, <successor>, <partof>, <pegging>, <tracking>, <interval> and <changeover>.

“#query” to delete a subelement, not element data can be specified with specifying the newly set subelement. In such a case, the deletion of subelement must be described before specifying the subelement.

In the following example, all the items that has already been set as production items of operation A are deleted and then item X is set newly.

```
<pslx type=“request” id=“123” action=“x”/>
<operation name=“operation A” action=“revise”>
<produce item=“#query”/>
<produce item=“item X”/>
</operation>
</pslx>
```

The subelements, <lotsize> and <tasksize>, don’t come under any of the above cases. Even if <lotsize> and <tasksize> have some applicable subelements, the new contents replace the subelements after deleting the subelements.

Even if the subelement of the target element is corrected, the data isn’t corrected at deeper level than the subelement level. In short, the information in deeper tag level than the subelement is overwritten together with the specified subelement.

4.3. Reaction to Request in Server side

The server providing service must respond to all request messages in the PSLX form. However, when the specific service isn’t implemented, the message about the non-implementation can be returned as an error. When receiving the request message not defined as the PSLX specification, an error message must be returned.

The server receiving the request message to the implemented interface must return a response message or an exception message

after processing for the request.

When a receipt request is added to the request message, a receipt message must be sent earlier than these messages. However, when an error occurs before producing the receipt message, only the exception message is returned and so the receipt message can be omitted.

4.3.1. Response Message

A response message is used for returning the result of processing the sent request message as a response. When the request cannot be met because a user doesn't have authority, an exception message must be returned.

The type attribute value in a <pslx> tag must be set at "response" in a response message. The unique character string in the responding application must be set in id attribute. Moreover the value of id attribute set in a request message must be set in ref attribute. Finally, the fit interface name for action attribute of a request message must be set in the action attribute.

A receipt confirmation request can be added to a response message to confirm the message reaches the receiver. When the receipt attribute in a <pslx> tag is set at "true", the receipt confirmation is added to the response message. In case of synchronous communication, the receipt confirmation cannot be requested.

Assuming that the request message (ID="123") as below is received,

```
<pslx type="request" id="123" action="x"/>
```

The usual example of the response message to this message is as below.

```
<pslx type="responce" id="456" ref="123"/>
```

An error and the warning contents can be set in a response message with an <error> tag as occasion demands. When including an error, the error is a part of a request message, and when all processing is an error, the error must be returned with an exception message.

4.3.2. Receipt Message

4.3.2. Receipt Message

When a request message or a response message includes a receipt request, the application receiving the message must return a receipt message immediately. “receipt” must be set in the type attribute of a <pslx> tag in the receipt message. The unique character string on the sending application must be set in id attribute. The id attribute value of the received message must be specified in ref attribute. The action attribute value of the received message that is an interface name must be set in action attribute.

For example, in case that the contents of the received message are as below

```
<pslx type=“request” id=“aaa” receipt=“true” action=“x”/>
```

The receipt message to this is set as below.

```
<pslx type=“receipt” id=“bbb” ref=“aaa”/>
```

4.3.3. Exception Message

An exception message must be sent when rejecting the received message, when the service to the message isn’t implemented, and when the message contents has an error. In the exception message, “exception” must be set in the type attribute in a <pslx> tag. The unique character string on the sending application must be set in id attribute. The id value of the received message must be specified in ref attribute. The action attribute value of the received message that is an interface name must be set in action attribute.

The exception contents in an exception message are set with an <error> tag. The exception is an error and warning. The sevierity attribute value must be ”error” for error and the value must be ”warning” for warning. Multiple error information or warning information can be set. The exception message must have one and more error information at least.

For example, when the request message contents have an error, the exception message as below is sent.

```
<pslx type="exception" id="bbb" ref="aaa">
<error code="Z01006" severity="error">there isn't the applicable service x.</error>
</pslx>
```

4.4. Data Permanence

The data described with PSLX can be made permanent as a file, not on-line data exchange between applications. When making PSLX data permanent, the type attribute value in a <pslx> tag must be omitted or must be set at "file".

XML declaration, <?xml … ?> tag is required in the head of file in the permanent data. The target data must be described under a <pslx> tag and the type attribute value must be set at "file" or omitted. id, ref, action, receipt attributes and error elements must not be specified.

The permanent data has the form as below.

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<pslx type="file">
<profile name="aaa.pslx" author="Y.Nishioka" version="1.0" create="2002-09-14T02:00:00"/>
<!—set PSLX data here -->
</pslx>
```

4.5. APS Agent Control

4.5.1. initPlan

This interface initializes planning processing. When the data related with plan has already been set, the data are initialized and so only the contents set later are valid.

Interface name		initPlan	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	profile	Order management information	0	1	1

Interface name		initPlan	Division	Response	
Required	Element name	Explanation	min	max	Actual
	Error	Error information	0	No	1

The contents of request message are as below.

```
<pslx type="request" id="a001" action="initPlan"/>
```

When specifying the problem name with a <profile> tag at the point of initializing, the applicable problem name data must be ready. If there is no applicable problem, a warning message must be returned and the problem of default set on the server must be ready.

After completing processing, a response message is returned. The response message is as below.

```
<pslx type="response" id="b001" ref="a001" action="initPlan"/>
```

4.5.2. makePlan

This interface makes planning executed. Planning indicates that the required parameter value is gotten based on the calculation model set beforehand as the parameter information or the numerical expression information.

Interface name		makePlan	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	profile	Order management information	0	1	1
	parameter	Information for plan	0	No	2
	expression	Information for plan	0	No	2

Interface name		makePlan	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error information	0	No	1
	parameter	Information for plan	0	No	1
	expression	Information for plan	0	No	2

When the problem name is set in a <profile> tag and also the problem isn't ready, planning is executed after setting the data equivalent to the problem name beforehand.

```
<pslx type="request" id="a001" action="makePlan">
<profile name="file name"/>
</pslx>
```

The target for planning can be limited and executed. For example, planning can be executed only with the applicable data by specifying the specification information set in the <expression> tag in the numerical expression information. The following example is the request message to execute "optimization model 1 ".

```
<pslx type="request" id="a001" action="makePlan">
<expression>
<spec name="model"><char valuel="optimization model  1 "/></spec>
</expression>
</pslx>
```

When solving the model consisting of constraint or evaluation, the result returned in a response message is the value of constraint or evaluation. This value is specified at the value attribute value or the violation attribute value in an <expression> tag.

If the <parameter> tag indicating the parameter information is specified in a request message and shows the calculation result, these parameter values must be calculated again. And also when the calculation parameter is set in constraint or evaluation, the values

must be calculated again similarly.

The following example shows that the contents of index 1 and index 2 are recalculated based on the newest information at that time.

```
<pslx type="request" id="a001" action="makePlan">
<parameter name="index 1 "/>
<parameter name="index 2 "/>
</pslx>
```

In setting planning items, the target condition can be specified like a plan inquiry. The following example is to recalculate the index on through-put.

```
<pslx type="request" id="a001" action="makePlan">
<parameter name="#query">
<spec name="type"><char name="through-put"/></spec>
</parameter>
</pslx>
```

In the response message to the request of planning, if the plan is normally executed and then the solution is gotten, the contents (parameter value) are returned. For example, the result of recalculating the index is as below.

```
<pslx type="response" id="a001" ref="a001" action="makePlan">
<parameter name="index 1 " value="2300"/>
<parameter name="index 2 " value="456000"/>
</pslx>
```

4.5.3. initSchedule

This interface clears the information of scheduler and restores the information to the initial state.

Interface name		initSchedule	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	profile	Order management information	0	1	1

Interface name		initSchedule	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error information	0	No	1

The message is as below.

```
<pslx type="request" id="a001" action="initSchedule"/>
```

The applicable data can be set as an initial scheduler by specifying the target scheduling problem to the scheduler.

```
<pslx type="request" id="a001" action="initSchedule">
<profile name="file name"/>
</pslx>
```

When receiving the request message to initialize scheduling, the response message about processing completion must be returned after initializing.

```
<pslx type="response" id="b001" ref="a001" action="initSchedule"/>
```

4.5.4. makeSchedule

This interface makes a scheduler execute scheduling. When the schedule has already been set, the data are rescheduled.

Interface name		makePlan	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	profile	Order management information	0	1	1
	parameter	Information for plan	0	No	2
	expression	Informaion for plan	0	No	2

Interface name	makePlan	Division	Response

Required	Element name	Explanation	min	max	Actuacl
	error	Error information	0	No	1
	parameter	Information for plan	0	No	1
	expression	Information for plan	0	No	2

When specifying operation, resource, item or order in a request message, the contents are rescheduled. If any element isn't set as the contents of the request message, the order or the operation that are not decided at that time are rescheduled.

When the planning period is set in a <profile> tag as the profile information, the period is for scheduling.

When a file name is specified as a profile and no data is read in a scheduler or the name is different from the present file name, the new information is read in.

```
<pslx type="request" id="a001" action="makeSchedule">
<profile name="file name"/>
</pslx>
```

The value is specified as below to reschedule only the operation meeting the specific conditions. In the following example, the operation that isn't fixed on resource Z is rescheduled.

```
<pslx type="request" id="a001" action="makeSchedule">
<operation name="#query">
<assign resource="Z"/>
<spec name="fixed"><char value="false"/></spec>
</operation>
</pslx>
```

When rescheduling normally, the ID of recalculated operation is returned as a result of schedule. When specifying a <query> tag in a request message, the subelement contents set in select attribute can be included in the result.

For example, the start date and the end date can be returned by

specifying ”start” and ”end”. When specifying that the start date and the end date are returned, the following message is returned.

```
<pslx type=“response” id=“b001” ref=“a001” action=“makeSchedule”>
<operation name=“S001”>
<start><time value=“2003-05-02T09:00:30“/></start>
<end><time value=“2003-05-02T12:00:30”/></end>
</operation>
<operation name=“S004”>…
</pslx>
```

4.6. Planning Information

4.6.1. setCalculation

This interface sets various parameters and the numerical expression information required for the plan. The parameter information is specified with a <parameter> tag and the numerical expression information is specified with an <expression> tag.

Interface name		setCalculation	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	profile	Order management information	0	1	1
	parameter	Information for plan	0	No	2
	expression	Information for plan	0	No	3

Interface name		setCalculation	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error information	0	No	1
	parameter	Processing confirmation	0	No	2
	expression	Processing	0	No	3

		confirmation			

A <parameter> tag is used for specifying the calculation method of parameter. In the following example, the latest date in the end dates of operation is defined as index 1.

```
<pslx type="request" id="a001" action="setCalculation">
<parameter name="index 1" action="create" operator="latest">
<operation name="#query" master="false">
<query select="end"/>
</operation>
</parameter>
</pslx>
```

As a result of the above request message, the ID of registered element is returned as a response message as below.

```
<pslx type="response" id="b001" ref="a001" action="setCalculation">
<parameter name="index 1" action="create"/>
</pslx>
```

The numerical expression model can also be defined by defining the numerical expression information such as constraint and evaluation. The constraint and the evaluation are expressed with an <expression> tag and are identified with the type attribute value. The type attributes, EQ, GE, LE, GT and LT, indicate the constraint The type attributes, MAX, MIN, indicate the evalucation.

When correcting the constraint or the evaluation, the numerical expression information of the specified ID is always replaced in a lump. The items, subelements of the numerical expression information, and the parameter information cannot be corrected partially.

When receiving the request message to newly set, correct and delete the constraint or the evaluation, the ID of the constraint or the evaluation that the processing succeeds are returned with a response message. The following is the response message showing that three expressions, E1, E2 and E3 are deleted to the deletion request.

```
<pslx type="response" id="b001" ref="a001" action="setConstraint">
<expression name="E1" action="delete"/>
<expression name="E2" action="delete"/>
<expression name="E3" action="delete"/>
</pslx>
```

4.6.2. getCalculation

This interface inquires the calculation method such as the parameter and the numerical expression information that have already been set. A <parameter> tag or an <expression> tag is used for inquiring the calculation method. However, a <query> tag cannot be set in the <parameter> tag of the calculation method. When inquiring the data, all the data are always returned.

Interface name		getCalculation	Division	Request	
Required	Element name	Explanation	min	max	Actial
	spec	For application extension	0	No	1
	profile	Management information	0	1	1
	parameter	Information for inquiry	0	No	2
	expression	Information for inquiry	0	No	3

Interface name		getCalculation	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error information	0	No	1
	parameter	Inquiry result	0	No	2
	expression	Inquiry result	0	No	3

```
<pslx type="request" id="a001" action="getCalculation">
<parameter name="index 1" />
<parameter name="index 2" />
</pslx>
```

The above example is the request message to inquire the contents of index 1 and index 2. And also the contents can be inquired with limiting the range as below without directly specifying the inquired index.

```
<pslx type="request" id="a001" action="getCalculation">
<parameter name="#query" >
<priority><min value="90"/></priority>
</parameter>
</pslx>
```

When inquiring the definition contents of constraint and evaluation, an <expression> tag is specified. For inquiring the contents, all the constraint and the evaluation making the model are inquired by specifying a model name in the specification besides directly specifying the ID of expression.

When the expression with the specific parameter x (definite constant) is inquired, the request message is created as below.

```
<pslx type="request" id="a001" action="getCalculation">
<expression name="#query">
<op parameter="x"/>
</expression>
</pslx>
```

All ways of calculating are returned to the inquiry about the paramter information and the numerical expression information with a response message.

In the former example, the information of E1 (evaluation) and E2 (constraint) including constant x is returned as below.

```
<pslx type="response" id="b001" ref="a001" action="getCalculation">
<expression name="E1" type="MAX">
<op no="1" operator="+" value="-2"/>
<op no="2" operator="*" parameter="X"/>
<op no="3" operator="+" parameter="Y"/>
</expression>
<expression name="E2" type="LE" rhd="4">
<op no="1" parameter="X"/>
</expression>
</pslx>
```

4.7. Share and Federation in Planning

4.7.1. setPlan

This interface sets the values of various parameters in planning. Plan parameter includes the constant data for plan besides the plan value. A <parameter> tag is used for setting the parameter value.

An <expression> tag is used for setting the value of calculation expression.

Interface name		setPlan	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	profile	Management information	0	1	1
○	parameter	Setting information	0	No	1
	expression	Setting information	0	No	1

Interface name		setPlan	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error information	0	No	1
	parameter	Setting result（only ID）	0	No	1
	expression	Setting result（only ID）	0	No	1

These interfaces are different from setCalculation and cannot set a calculation method and the constraint contents.

The value attribute can be set with the <parameter> tag indicating parameter. The type attribute value of the parameter information can be changed. If this value is a constant, the value attribute value may be changed by executing planning.

The following example is the request message to set the target value of “May production quantity” at 2,000 as a production plan value.

```
<pslx type="request" id="a001" action="setPlan">
<parameter name="May production quantity" value="2000" unit="piece" action="revise"/>
</pslx>
```

The right side information in the numerical expression information can be set with an <expression> tag. This value is a constant in constraint or the target value in evaluation. This is specified as a rhd attribute value.

The following example is the request message to set the value of index, "this month profit". The type attribute of this expression is "MAX".

```
<pslx type="request" id="a001" action="setPlan">
<expression name="this month profit" rhd="504500" unit="yen" action="revise"/>
</pslx>
```

The calculation expression data, <expression> tag cannot be added newly. In case of parameter <parameter>, only the parameter value can be set but the calculation method cannot be specified. The data cannot be deleted in both the parameter information and the numerical expression information.

4.7.2. getPlan

This interface inquires parameter values or constraint data values (evaluation index). The value attribute is inquired with the <parameter> tag indicating parameter. The values of value attribute, rhd attribute and violation attribute can be inquired with the <expression> tag indicating the numerical expression information.

Interface name		getPlan	Division	Request	
Required	Element name		min	max	Actual
	Spec	For application extension	0	No	1
	profile	Management information	0	1	1
○	parameter	Contents to be inquired	0	No	1

	expression	Contents to be inquired	0	No	1

Interface name		getPlan	Division	Response	
Required	Element name		min	max	Actual
	error	Error contents	0	No	1
	parameter	Inquiry result	0	No	1
	expression	Inquiry result	0	No	1

The example of inquiry message is as below.

```
<pslx type="request" id="a001" action="getPlan">
<expression name="this month profit"/>
</pslx>
```

The <query> tag specifying the details of the inquired element data cannot be used. Therefore all attributes such as value attribute, rhd attribute, violation attribute and unit attribute of the applicable constraint, evaluation and parameter are returned.

4.7.3. setSchedule

This interface sends the contents created by scheduling to the party. This interface is used for creating a new schedule or changing the existing schedule. Multiple operation changes can be sent at the same time.

Interface name		setSchedule	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	unit	Appendix informaion	0	No	3
	scale	Appendix information	0	No	2
	profile	Management information	0	1	1

	item	Setting contents	0	No	1
	resource	Setting contents	0	No	1
	lot	Setting contents	0	No	1
	task	Setting contents	0	No	1
○	event	Setting contents	0	No	2
	operation	Setting contents	0	No	1
	order	Setting contents	0	No	1

Interface name		setSchedule	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error information	0	No	1
	operation	Setting result（only ID）	0	No	

The target operation cannot include the master (master="true") information.

For example, when the plan date of operation is changed, the following message is sent to the receiving agent. Only the changed operation is sent.

```
<pslx type="request" id="a001" action="setSchedule">
<operation name="P001" action="revise">
<start><time value="2003-05-03T14:30:00"/></start>
<end><time value="2003-05-03T15:50:00"/></end>
</operation>
</pslx>
```

When an operation is produced newly, the start and end dates and the assigned resouce are specified as below.

```
<pslx type="request" id="a001" action="setSchedule">
<resource name="Z002"/>
<operation name="P002" action="create">
<start><time value="2003-05-03T16:30:00"/></start>
<end><time value="2003-05-03T16:50:00"/></end>
<assign ref="Z002">
</operation>
</pslx>
```

When all the data of the start date, the end date and the processing time aren't changed at the same time according to the change of schedule contents, the relation among them may not stand up.

An <assign> tag is specified for changing the information about the used resource. When the newly specified operation information in which the nth attribute value is the same as the sel attribute value is already used, the new contents replace the former information. If the data doesn't have new contents, the assignment information is set newly.

The list of the operations accepting setting is returned as a response message for setting schedule contents. In the list, only the operation IDs are expressed with name attribute and other data aren't sent back. For example, the response message to the above request message is as below.

```
<pslx type="response" id="b001" ref="a001" action="setSchedule">
<operation name="P002"/>
</pslx>
```

The independent event information besides the operation information can be set for schedule.

4.7.4. getSchedule

This interface inquires the contents of schedule. Mainly the operation information and the event information (independent event) are inquired for schedule. The related element data can be inquired if necessary.

Interface name		getSchedule	Division	Request	
Required	Element name		min	max	Actual
	spec	For application extension	0	No	1
	profile	Management information	0	1	1
○	event	Inquiry contents	0	No	2
	operation	Inquiry contents	0	No	1

Interface name		getSchedule	Division	Response	
Required	Element name		min	max	Actual
	error	Error contents	0	No	1
	unit	Appendix information	0	No	3
	scale	Appendix information	0	No	2
	event	Inquiry result	0	No	2
	operation	Inquiry result	0	No	1

When inquiring the start date, the end date and the assigned resource with the operation information, the values are set as below.

```
<pslx type="request" id="a001" action="getSchedule">
<operation name="#query">
<query select="attribute"/>
<query select="start"/>
<query select="end"/>
<query select="assign"/>
</pslx>
```

The contents of the schedule data returned with a response message is as below.

```
<pslx type="response" id="b001" ref="a001" action="getSchedule">
<resource name="Z001"/>
<operation name="P001">
<start><time value="2003-05-03T12:30:00"/></start>
<end><time value=" 2003-05-03T16:10:00"/></end>
<assign resource="Z001"/>
</operation>
</pslx>
```

The event data that is not the master information can also be inquired for schedule.

4.8. Master Data

4.8.1. setParty

This interface registers the master information on customer and supplier. The information is registered with a specification <spec> because the tags indicating a customer or a supplier aren't prescribed

in detail. The messagea to be sent are as below.

Interface name		setParty	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	profile	Management information	0	1	1
○	customer	Setting contents	0	No	1
	supplier	Setting contents	0	No	1

Interface name		setParty	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error information	0	No	1
	customer	Setting result (only ID)	0	No	1
	supplier	Setting result (only ID)	0	No	1

```
<pslx type="request" id="a001" action="setParty">
<customer name="ABC store" action="create">
<address name="Koganei 001"/>
<spec name="address1"><char value="184-8584"/></spec>
<spec name="phone"><char value="012-345-6789"/></spec>
<spec name="fax"><char value="012-345-6789"/></spec>
<spec name="URL"><char value="http://www.img.k.hosei.ac.jp"/></spec>
<spec name="address1"><char value="3-7-2, Kajino-cho, Koganei City, Tokyo"/></spec>
</customer>
</pslx>
```

When the registaration is executed normally, only the IDs of the registered data are returned with a response message. For example, the following message is returned.

```
<pslx type="response" id="b001" ref="a001" action="setParty">
<customer name="ABC store"/>
</pslx>
```

4.8.2. getParty

This interface inquires the master information on customer and supplier.

Interface name		getParty	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	profile	Management information	0	1	1
○	customer	Contents to be inquired	1	No	1
	supplier	Contents to be inquired	1	No	1

Interface name		getParty	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error contents	0	No	1
	customer	Inquiry result (configuration data)	1	No	1
	supplier	Inquiry result (configuration data)	1	No	1

When getting the information about the customer whose person in charge of sales is Nishioka, the request message as below is created. In such a case, the ID and telephone No. of the applicable customer are returned.

```
<pslx type="request" id="a001" action="getParty">
<customer name="#query">
<spec name="person in charge of sales"><char value="Nishioka"/></spec>
<query select="phone"/>
</customer>
</pslx>
```

The target data meeting the conditions are set in a response message and are returned as below.

```
<pslx type="response" id="b001" ref="a001" action="getParty">
<supplier name="A Trading Company">
<spec name="phone"><char value="012-345-678"/></spec>
</supplier>
<supplier name="B Industry">
<spec name="phone"><char value="023-456-789"/></spec>
</supplier>
</pslx>
```

4.8.3. setProduct

This interface registers the master information of product. The information on product is mainly set with an item <item> tag. However, when expressing Manufacturing BOM, the relation between items can be set through the operation, not directly. The customer information to product or the supplier information to material can be set as the appendix information of item.

Interface name		setProduct	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	unit	Appendix information	0	No	3
	profile	Management information	0	1	1
○	item	Information to be set	0	No	1
	resource	Information to be set	0	No	3
	event	Information to be set	0	No	3
	operation	Information to be set	0	No	2

Interface name		setProduct	Division	Response	
Required	Element	Explanation	min	max	Actual

	name				
	error	Error information	0	No	1
	item	Setting result（only ID）	0	No	1

The request message as below is sent for registering the product master information.

```
<pslx type="request" id="a001" action="setProduct">
<item name="product A" action="create">
<price value="15000"/>
<spec name="size"><qty value="500" unit="mm"/></spec>
<spec name="color"><char value="blue"/></spec>
</item>
</pslx>
```

When the product master information is registered newly and is corrected normally, the ID list of applicable items is returned. A response message is as below.

```
<pslx type="response" id="b001" ref="a001" action="setProduct">
<item name="product A"/>
</pslx>
```

The time series information such as stock and capacity, item subelements, cannot be set as the master information. These data are separately set with the interface such as setStock or setCapacity.

4.8.4. getProduct

This interface inquires the product information. With getProduct, the contents of individual products can be inquired by specifying IDs of the products and also the information of the applicable product can be inquired by giving constraint.

Interface name		getProduct	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	profile	Management information	0	1	1

○	item	Contents to be inquired	1	No	1

Interface name		getProduct	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error contents	0	No	1
	unit	Appendix information	0	No	3
	item	Inquiry result	0	No	1
	operation	Inquiry result (in case of BOM)	0	No	3

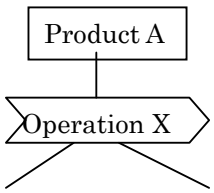
For example, the request messge as below is created for inquiring the material bought from supplier A.

```
<pslx type="request" id="a001" action="getProduct">
<item name="#query">
<spec name="supplier"><char value="supplier A"/></spec>
</item>
</pslx>
```

The target data meeting the conditions are set in a response message and are returned as below.

```
<pslx type="response" id="b001" ref="a001" action="getProduct">
<item name="material A"/>
<item name="material B"/>
<item name="material C"/>
</pslx>
```

Manufacturing BOM information consists of the item information and the operation information as follows. For instance, the following figure shows that Manufacturing BOM consisting of product A, parts B and parts C is set through operation X.



Parts B　　Parts C

In the above case, the hierarchy seen from a viewpoint of product A is specified in the depth attribute value in a <query> tag for inquiring parts B and Parts C as a child of product A.

```
<pslx type="request" id="a001" action="getProduct">
<item name="product A">
<query select="produce" length="2"/>
</item>
</pslx>
```

In the hierarchy seen from a viewpoint of product A , length 1 corresponds to the operation X layer and length 2 corresponds to the layer of parts B and parts C consumed by operation X.　Therefore the response message to the above request message is as below.

```
<pslx type="response" id="b001" ref="a001" action="getProduct">
<item name="product A"/>
<item name="parts B"/>
<item name="parts C"/>
<operation name="operation X">
<produce item="product A"/>
<consume item="parts B"/>
<consume item="parts C"/>
</operation>
</pslx>
```

4.8.5. setProcess

This interface registers the process information.　The process information is expressed maily with resource and operation.　In case of the operation information, only what the master attribute is "true" is inquired.

Interface name		setProcess	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	unit	Appendix information	0	No	3
	profile	Management	0	1	1

		information			
	item	Information to be set	0	No	3
	resource	Information to be set	0	No	1
	event	Information to be set	0	No	2
	operation	Information to be set	0	No	1

Interface name		setProcess	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error information	0	No	1
	resource	Setting result（only ID）			
	event	Setting result（only ID）	0	No	2
	operation	Setting result（only ID）	0	No	1

The following shows that the operation information and the resource information are set newly.

```
<pslx type="request" id="a001" action="setProcess">
<resource name="R04" action="create"/>
<operation name="P0001" action="create">
<assign resource="R04"/>
</operation>
</pslx>
```

The date information and the time series information such as load, capacity, start, end, duration and progress as subelements of operation and the resource cannot be set as the master information.

When the registration is executed normally, the list of the processed basic elements is returned. The following response message is returned to the above request message.

```
<pslx type="response" id="b001" ref="a001" action="setData">
<resource name="R04" action="create"/>
<operation name="P0001" action="create"/>
```

```
</pslx>
```

4.8.6. getProcess

This interface inquires the process information. The process information can be inquired for the master information of operation and the resource information. In case of the operation information, only what the master attribute in an <operation> tag is true is inquired.

Interface name		getProcess	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	profile	Management information	0	1	1
	resource	Contents to be inquired（query）	1	No	1
	event	Contents to be inquired（query）	0	No	2
	operation	Contents to be inquired（query）	1	No	1

Interface name		getProcess	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error contents	0	No	1
	unit	Appendix information	0	No	3
	resource	Inquiry result (configuration data)	0	No	1
	event	Inquiry result (configuration data)	0	No	2
	operation	Inquiry result	0	No	1

		(configuration data)			

In the following example, the resource used by the operation that can produce product A is inquired.

```
<pslx type="request" id="a001" action="getProcess">
<operation name="#query">
<produce item="product A"/>
<query select="assign"/>
</operation>
</pslx>
```

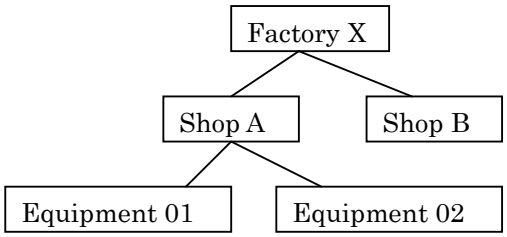
In the response message to the above request message, the applicable operation is returned adding an <assign> tag as below.

```
<pslx type="response" id="b001" ref="a001" action="getProcess">
<operation name="P0004"><assign resource="resource B"/></operation
<operation name="P0007"><assign resource="resource B"/></operation
<operation name="P0011"><assign resource="resource C"/></operation
</pslx>
```

The hierarchical structure of operation or resource can be inquired using a <partof> tag. In the following example, the hierarchical relation of Factory X is inquired. Depth attribute shows what place of the hierarchy is traced to and in case that the value is minus, the relation of partof is traced conversely (from the whole to the part).

```
<pslx type="request" id="a001" action="getProcess">
<operation name="factory X">
<query select="partof" depth="-2"/>
</operation>
</pslx>
```

For example, when the hierarchical structure of Factory X is as the following figure, the response message to the above request is as below.




```
<pslx type="response" id="b001" ref="a001" action="getProcess">
<resource name="factory X"/>
<resource name="shop A"><partof resource="factory X"/></resource>
<resource name="shop B"><partof resource="factory X"/></resource>
<resource name="equipment 01"><partof resource="shop A"/></resource>
<resource name="equipment 02"><partof resource="shop A"/></resource>
</pslx>
```

The date information and the time series information such as load, capacity, start, end, duration and progress as subelements of operation or resource cannot be inquired as the master information. These data are inquired with another type of request message.

4.9. Order Information

4.9.1. setOrder

This interface registers, corrects or deletes order. The item and the operation information referred to in the order information can be operated at the same time as the order specific information. The order specific information and the available information in the whole defined as a master must be able to be distinguished on the server application.

Interface name		setOrder	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	unit	Unit definition	0	No	3

	scale	Scale unit	0	No	2
	profile	Order management information	0	1	1
	customer	In case of customer order	0	No	1
	supplier	In case of purchase order			2
	item	In case of item order	0	No	1
	resource	In case of resource order			3
	operation	In case of operation order			2
○	order	Order contents	1	No	1

Interface name		setOrder	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error information	0	No	1
○	order	Setting result （only ID）	1	No	1

The following shows a general order example.

```
<pslx type="request" id="a001" action="setOrder">
<order name="K001" item="product A" action="create"><qty value="10"/></order>
<order name="K002" item="product B" action="create"><qty value="30"/></order>
</pslx>
```

Usually, when the receiver of the order information has the item information, the item information isn't written down with the message. When individual products are different like cutom-made articles, the item information is written down. The example of the order for a custom-made article is as below.

```
<pslx type="request" id="a002" action="setOrder">
<item name="custom-made article">
```

```
<spec name="drawing"><char value="http://abc.com/3d-data/0123.cad"/></spec>
</item>
<order name="K003" item="custom-made article" action="create"><qty value="1"/></order>
</pslx>
```

The receiving application must process the specified order and must return the list of accepted orders as a result to this request message. The message is as below.

```
<pslx type="response" id="b001" ref="a001" action="setOrder">
<order name="K001"/>
<order name="K002"/>
</pslx>
```

When not accepting a part or all of orders, the order is shown with the warning message as below. The following example shows that K002 is accepted but K001 isn't accepted.

```
<pslx type="response" id="b001" ref="a001" action="setOrder">
<error code="W00123" severity="warning">the accept of order "K001" was rejected.</error>
<order name="K002"/>
</pslx>
```

While, the contents of the order already sent and accepted can be changed later. When changing the order, the action attribute value in an <order> tag is set at "revise". Only the part set as an attribute or an element is updated.

```
<pslx type="request" id="a001" action="setOrder">
<order name="K002" action="revise"><qty value="20"/></order>
</pslx>
```

The contents of the item data, the resource data and the operation data referred to in the order information can be changed as the appendix information of order. However, when those data are registered as the master information, the master information isn't changed and is updated as the individual information for the order. Therefore another order information referring to the item doesn't affect the data.

4.9.2. getOrder

This interface inquires the contents of order. The information on all

the orders or the order fulfilling the specific conditions can be gotten out of the order information on the server side.

Interface name		getOrder	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	profile	Management information	0	1	1
○	order	Inquiry contents（query）	1	No	1

Interface name		getOrder	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error contents	0	No	1
	unit	Unit definition	0	No	3
	scale	Scale unit	0	No	2
	order	Inquiry result（configuration data）	0	No	1

If only the order that a customer name is ”ABC Trading Company” and the price is 100,000 yen and more is taken out, setting is as below.

```
<pslx type=“request” id=“a001” action=“getOrder”>
<order name=“#query” customer=“ABC trading comapny”>
<price><min value=“100000”/></price></order>
</pslx>
```

In the above case, the list in which only order names are set is returned as a response message.

When the concrete contents of order are wanted, the required information must be shown with a <query> tag individually. In the following example, the customer ID, the item ID, the quantity and the price of the order are inquired.

```
<pslx type="request" id="a001" action="getOrder">
<order name="#query">
<query select="attribute"/>
<query select="qty"/>
<query select="price"/>
</order>
</pslx>
```

The response message with the applicable data is returned as a result of inquiring order as below.

```
<pslx type="response" id="b001" ref="a001" action="getOrder">
<order name="K001" customer="ABC trading company" item="product A">
<qty value="200"/><price value="52000" unit="yen"/>
</order>
</pslx>
```

✧ **Inquiring the duetime information**

Select attribute is set at "duetime" with a <query> tag for inquiring the duetime. And so the duetime information of applicable order is returned. In the following example, the duetime information is returned to all the orders from ABC Trading Company.

```
<pslx type="request" id="a001" action="getOrder">
<order name="#query" customer="ABC trading company"/>
<query select="duetime"/>
</pslx>
```

The duetime information returned as a response message is always the newest duetime information. When the status attribute value in an <order> tag is set at "fixed", the replied duetime is regarded as being promised firmly.

```
<pslx type="response" id="b001" ref="a001" action="getOrder">
<order name="K001" status="fixed">
<duetime><time value="2003-05-02T13:00:00"/></duetime>
</order>
</pslx>
```

Pegging (pegging) can be inquired as well. In such a case, the status where some orders are associated through a <pegging> tag is set in a response message.

4.9.3. setEstimation

This interface sets the order estimation result. In case of the order

estimation result, a flag, “temporal”, is set in the order and the order vanishes automatically after the period defined by each application passes.

Interface name		setEstimation	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	unit	Unit definition	0	No	3
	scale	Scale unit	0	No	2
	profile	Management information	0	1	1
○	order	Estimation	1	No	1

Interface name		SetEstimation	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error information	0	No	1
	order	Estimation result	1	No	1

The estimated contents are set in the party for setting the order estimation. The specified contents are expressed with an <order> tag. However, status=”temporal” must specify that the order isn’t an official order.

The contents of order estimation are the duetime available to supply and the quantity available to supply. The sample is shown as follows.

```
<pslx type=“request” id=“a001” action=“setEstimation”>
<order name=“K123”>
<duetime><time value=“2003-05-02T15:00:00”/></duetime>
<qty value=“2000” unit=“piece”/>
</order>
</pslx>
```

The item list of the specified estimation is returned with the response message for setting the estimation information.

```
<pslx type=“response” id=“b001” ref=“a001” action=“setEstimation”>
<order name=“K123”/>
```

```
</pslx>
```

4.9.4. getEstimation

This interface requests the order estimation. The duetime and the quantity available to supply in case of ordering before formally ordering are inquired. The specified order is different from the usual order and is deleted after the server calculates the estimation information. In short, the order is set at status="temporal" in an <order> tag.

Interface name		getEstimation	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	unit	Unit definition	0	No	3
	scale	Scale unit	0	No	2
	profile	Order management information	0	1	1
	customer	In case of customer order	0	No	1
	supplier	In case of purchase order			2
	item	In case of item order	0	No	1
	resource	In case of resource order			3
	operation	In case of operation order			2
○	order	Order contents	1	No	1

Interface name		getEstimation	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error information	0	No	1

	order	Inquiry contents of estimation	1	No	1

For example, when 2,000 of product A are wanted, the order duetime is inquired as below.

```
<pslx type="request" id="a001" action="getEstimation">
<order name="#query" item="product A" status="temporal">
<qty value="2000" unit="piece"/>
<query select="duetime"/>
</order>
</pslx>
```

The order name specified in this tag is "#query" and the data is inquired in the same form as the usual data inquiry. However, the applicable order actually existing on the server isn't retrieved. The order is temporally produced on the server and then the contents is returned. This is judged by specifying status="temporal". The multiple kinds of the required orders can be set at the same time.

For instance, the application receiving this request message needs to execute the procedure that is producing the applicable order information with status="temporal", rescheduling, returning the order information as a scheduling result with a response message and restoring the order to the status before rescheduling after deleting the temporally added order.

The response message to the request message for the order estimation shows the answer as a result of estimating the requested order contents. The answer contents are the duetime and the quantity available to supply.

```
<pslx type="response" id="b001" ref="a001" action="getEstimation">
<order name="K123">
<duetime><time value="2003-05-02T15:00:00"/></duetime>
<qty value="2000" unit="piece"/>
</order>
</pslx>
```

4.10. Progress Information

4.10.1. setProgress

This interface sets the progress information to operation or order.

Interface name		setProgress	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	unit	Unit definition	0	No	3
	scale	Scale unit	0	No	2
	profile	Management information	0	1	1
○	operation	Progress information to be set	0	No	1
	order	Progress information to be set	0	No	1

Interface name		setProgress	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error information	0	No	1
	operation	Setting result（only ID）	0	No	1
	order	Setting reruslt（only ID）	0	No	1

The example is shown as below.

```
<pslx type="request" id="a001" action="setProgress">
<operation name="P012">
<progress status="completed">
<time value="2003-05-01T19:00:00"/></progress>
</operation>
</pslx>
```

For deleting the progress information, #init is set in the date part in a

progress tag of the deleted order tag or an operation tag, and so a <qty> tag isn't set. In the following example, all the progress data set in operation "P012" are deleted.

```
<pslx type="request" id="a001" action="setProgress">
<operation name="P012" action="revise">
<progress><time ref="#init"/></progress>
</operation>
</pslx>
```

The ID of the set operation or the set order is returned for responding to the progress information setting.

```
<pslx type="response" id="b001" ref="a001" action="setProgress">
<operation name="P012" action="revise"/>
</pslx>
```

4.10.2. getProgress

This interface inquires the progress information. The progress information can be inquired in the operation information and the order information. The operation information or the order information to be inquired is specified as an element of <pslx> tag. When some progress data are set in one element, all the data are returned.

Interface name		getProgress	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	profile	Mangement information	0	1	1
○	operation	Target to be inquired	1	No	1
	order	Target to be inquied	1	No	1

Interface name		getProgress	Division	Response	
Required	Element name	Explanation	min	max	Actual

	error	Error contents	0	No	1
	unit	Unit definition	0	No	3
	scale	Scale unit	0	No	2
	operation	Inquiry result（progress data）	0	No	1
	order	Inquiry result（progress data）	0	No	1

The inquired contents are always a <progress> tag. So setting select ="progress" with a <query> tag can be omitted.

```
<pslx type="request" id="a001" action="getProgress">
<order name="#query" customer="ABC trading company"/>
</pslx>
```

In the above example, the progress of all the orders from ABC Trading Company is inquired. The response message to this inquiry is as below.

```
<pslx type="response" id="b001" ref="a001" action="getProgress">
<order name="K004"><progress status="completed"/></order>
<order name="K015"><progress status="ready"/></order>
<order name="K021"><progress status="created"/></order>
</pslx>
```

4.11. Stock Information

4.11.1. setStock

This interface sets the stock information. This interface is used for setting the stock information to the specific item. When the stock information is set as the information of later date than the date specified before, the new information is added with leaving the former information. When the newly set date is earlier than the already set date, the stock information already set is deleted and overwritten.

Interface name		setStock	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	unit	Appendix	0	No	3

		information			
	scale	Appendix information	0	No	2
	profile	Management information	0	1	1
○	item	Setting information（stock information）	0	No	1

Interface name		setStock	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error information	0	No	1
	item	Setting result（only ID）	0	No	1

The sample to set the stock data is shown as below.

```
<pslx type="request" id="a001" action="setStock">
<item name="K001">
<stock><time ref="#init"/><qty value="20"/></stock>
<stock><time value="2003-05-02T00:00:00"/><qty value="30"/></stock>
</item>
</pslx>
```

If a stock tag exists in an item element, the stock data of the element is deleted. When setting the stock information without specifying the date information, the omitted value of date is ”#init” and so the information newly set replaces the information already set.

Setting the stock value is always correcting the item information. When the specified item doesn’t exist, it is an error because it is a premise that action attribute in an <item> tag is ”revise”.

When the relative attribute in the tag <stock> indicating the stock value is ”true”, the shown value is a relative value and shows increase and decrease of the stock value from the just-before value. In such a case, even if the just-before data is the same date, the data isn’t

deleted. In other cases, all the stock information on the set date and the later dates are deleted.

4.11.2. getStock

This interface inquires the stock information. The action attribute value in a <pslx> tag is “getStock” for inquiring the stock information of the specific item. The following shows the example to inquire the stock information in a request message. In this example, the ID of inquired item is directly specified. The contents to be inquired are always a <stock> tag and so must not be set at select=”stock” with a <query> tag.

Interface name		getStock	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	profile	Management information	0	1	1
○	item	Contents to be inquired	1	No	1

Interface name		getStock	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error contents	0	No	1
	unit	Unit definition	0	No	3
	scale	Scale unit	0	No	2
	item	Inquiry result（stock data）	0	No	1

```
<pslx type=“request” id=“a001” action=“getStock”>
<item name=“product X”/>
</pslx>
```

The following shows the contents of stock level of the applicable

product returned in time series as a response message example to this message.

```
<pslx type="response" id="b001" ref="a001" action="getStock">
<item name="product X">
<stock><time value="2003-05-01T00:00:00"/><qty value="2000"/></stock>
<stock><time value="2003-05-02T00:00:00"/><qty value="1800"/></stock>
<stock><time value="2003-05-03T00:00:00"/><qty value="1300"/></stock>
</item>
</pslx>
```

The start attribute and the end attribute in a <query> tag are used for inquiring the stock level at the specific date. The default values of start attribute and end attribute are #init and #final respectively. Therefore the stock value of May 2 and later can be gotten by setting as below.

```
<pslx type="request" id="a001" action="getStock">
<item name="#query">
<query select="stock" start="2003-05-02T00:00:00"/>
</item>
</pslx>
```

In the information returned as a response message, the relative attribute in a <stock> tag is always "false" and is expressed with a absolute value.

4.12. Load Information

4.12.1. setLoad

This interface sets the load value to resource in time series. When the load information has already been set in the specified resource, the load information is newly added to those load data. However, when the same date or the later dates than the date in the added information exist already, those existing data are deleted. When the date in the newly set information is the same as the date in the already set information, setting the load information depends on the relative attribute value in a <load> tag. See "setting the stock information".

Interface name	setLoad	Division	Request

Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	unit	Unit definition	0	No	3
	scale	Scale unit	0	No	2
	profile	Management information	0	1	1
○	resource	Setting information（load information）	0	No	1

Interface name		setLoad	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error information	0	No	1
	resource	Setting result（only ID）	0	No	1

```
<pslx type="request" id="a001" action="setLoad">
<resource name="K001">
<load><time ref="#init"/><qty value="2"/></load>
<load><time value="2003-05-01T00:00:00"/></qty value="0"/></load>
</resource>
</pslx>
```

The list of resource with the set load information is shown to the response message for the request message to set the stock information. If no resource specified in a request message exists, an exception message is returned. When only a part of the resource doesn't exist, the load information of the existing resource is returned as a response message and the information of not-existing resource is returned with an <error> tag as the warning.

4.12.2. getLoad

This interface inquires the time series information of load to resource. All the load data of specified resource are returned. The following messages are used for inquiring the load information.

Interface name		getLoad	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	profile	Management information	0	1	1
○	resource	Contents to be inquired	1	No	1

Interface name		getLoad	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error contents	0	No	1
	unit	Unit definition	0	No	3
	scale	Scale unit	0	No	2
	resource	Inquiry result（load data）	0	No	1

```
<pslx type="request" id="a001" action="getLoad">
<resource name="K001"/>
<resource name="K003"/>
</pslx>
```

The above example request to inquire all the load data about the resources, K001 and K003. The contents of <load> tag are requested as an inquiry result and so essentially <query select="load"/> is needed but can be omitted because the load information is always inquired with getLoad.

In the response message to a request message, the load value of the inquired resource is set in time series and is returned.

```
<pslx type="response" id="b001" ref="a001" action="getLoad">
<resource name="K001">
<load><time value="2003-05-01T00:00:00"/><qty value="3"/></load>
<load><time value="2003-05-02T00:00:00"/><qty value="0"/></load>
<load><time value="2003-05-03T00:00:00"/><qty value="1"/></load>
</resource>
</pslx>
```

4.13. Capacity Information

4.13.1. setCapacity

This interface sets the capacity information. When directly setting the available capacity without using a calendar, the following message is sent. The capacity can be set in resource or item. Level 2 and over are required for implementing this interface.

Interface name		setCapacity	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	unit	Unit definition	0	No	3
	scale	Scale unit	0	No	2
	profile	Management information	0	1	1
○	item	Setting information（capacity information）	0	No	2
	resource	Setting information（capacity information）	0	No	2

Interface name		setCapacity	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error infomraiton	0	No	1
	item	Setting result（only ID）	0	No	2
	resource	Setting result（only ID）	0	No	2

In the following example, the capacity on May 1 is set at 200 and the capacity on May 2 is set at 100.

```
<pslx type="request" id="a001" action="setCapacity">
<resource name="machine A">
<capacity><time value="2003-05-01T00:00:00"/><qty value="200"/></capacity>
<capacity><time value="2003-05-01T00:00:00"/><qty value="100"/></capacity>
</resource>
</pslx>
```

When the capacity information is already set in the target resource, the capacity data is newly set with being added to the end of existing data. However, when there are the data with the same date or the later date than the date of set data, the data is deleted. When the relative attribute in a <capacity> tag is "true" and also the data has the same date, the former data isn't deleted.

When a <capacity> tag isn't set, all the capacity data are deleted.

4.13.2. getCapacity

This interface inquires the capacity information. The request message in which the action attribute in a <pslx> tag is set at "setCapacity" is used for inquiring the capacity information. The capacity information can be inquired in the resource and item that have the capacity information. Level 2 and over are required for implementing this interface.

Interface name		getCapacity	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	profile	Management information	0	1	1
○	item	Contents to be inquired	1	No	1
	resource	Contents to be inquired	1	No	1

Interface name		getCapacity	Division	Response	
Required	Element name	Explanation	min	max	Actual

	error	Error contents	0	No	1
	unit	Unit defintion	0	No	3
	scale	Scale unit	0	No	2
	item	Inquiry result（configuration data）	1	No	2
	resource	Inquiry result（configuration data）	1	No	2

In the following example, the capacity of all the resources on the first floor is inquired. In such a cae, the inquired contents are always <capacity> and so setting select="capacity" with a <query> tag can be omitted.

```
<pslx type="request" id="a001" action="getCapacity">
<resource name="#query"><partof resource="first floor"/></resource>
</pslx>
```

In the response message to inquire the capacity information, the capacity values of the inquired resource or item are shown in time series. All the time series data are returned except in the cases that the start attribute and the end attribute in a <query> tag are set or that the start attribute and the end attribute are set with a <profile> tag.

When the start attribute and the end attribute are set with the <query> tag set in the applicable item or resource and a <profile> tag, the time series information about the period corresponding to the product set of data is returned.

The example of response message is as follows.

```
<pslx type="response" id="b001" ref="a001" action="getCapacity">
<resource name="R001">
<capacity><time value="2003-05-02T00:00:00"/><qty value="4"/></capacity>
<capacity><time value="2003-05-03T00:00:00"/><qty value="0"/></capacity>
</resource>
</pslx>
```

4.13.3. setCalendar

This interface sets the calendar information. Setting the capacity information can be replaced with setting the calendar information. The calendar information is to simply set the time series information of capacity with the shift information that is a pattern of the capacity information. Therefore the calendar information must be set with specifying the shift information. However, when the shift information is shared with the receiver beforehand, this shift information can be omitted. Level 3 and over are required for implementing this interface.

Interface name		setCalendar	Division	Request	
Required	Element name	Explanation	min	max	Actual
	Spec	For application extension	0	No	1
	unit	Appendix information	0	No	3
	scale	Appendix information	0	No	2
	shift	Appendix information	0	No	3
	profile	Management information	0	1	1
○	item	Setting information（capacity information）	0	No	1
	resource	Setting information（capacity information）	0	No	1

Interface name		setCalendar	Division	Response	
Required	Element name	Explanation	min	max	Actual

	error	Error information	0	No	1
	item	Setting result (only ID)	0	No	1
	resource	Setting result (only ID)	0	No	1

```
<pslx type="request" id="a001" action="setCalendar">
<shift name="daily">
<capacity><time value="2000-01-01T00:00:00"/><qty value="5"/></capacity>
<capacity><time value="2000-01-01T10:00:00"/><qty value="0"/></capacity>
</shift>
<resource name="R001">
<calendar shift="daily" time="2003-05-01T08:00:00"/>
<calendar shift="daily" time="2003-05-02T08:00:00"/>
</resource>
</pslx>
```

In the above example, the change of capacity --- the capacity from 8:00 to 18:00 on May 1 is 5; the capacity from 8:00 to 18:00 on May 2 is 5; other capacities are 0 --- is expressed with the shift information, daily.

The application receiving the request to set the calendar information executes setting and returns the list of the set item or resource as a result with a response message. When there is an element, which cannot be set, the contents are described in a response message as the warning with an <error> tag.

4.13.4. getCalendar

This interface inquires the calendar informan. "getCalendar" is specified in the action attribute in a <pslx> tag for inquiring the calendar information. When inquiring the calendar information of resource, the ID is directly specified or the range is specified with "#query". Level 3 and over are required for implementing this interface.

Interface name		getCalendar	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	profile	Management	0	1	1

		information			
○	resource	Contents to be inquired	1	No	1

Interface name		getCalendar	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error contents	0	No	1
	unit	Unit definition	0	No	3
	scale	Scale unit	0	No	2
	shift	Inquiry result	0	No	3
	resource	Inquiry result	0	No	1

For instance, when the whole factory and each facility are defined with the <partof> relation, it is convenient to specify “whole factory”, the upper resource for inquiry.

```
<pslx type=“request” id=“a001” action=“getCalendar”>
<resource name=“R001”/>
</pslx>
```

The calendar information is set to each resource and is the time series information. The calendar information in all periods may be inquired or the inquired period may be restricted. When restricting the period, a <query> tag is set as an element of a <resource> tag and a start attribute and an end attribute are specified in the tag. And also the start attribute and the end attribute in a <profile> tag can be used for resctricting the period.

In the response message for inquiring the calendar information, the referred shift information is described besides the calendar information to resource.

4.14. Lot Information

4.14.1. setLot

This interface sets the lot information. The lot information is specified with a message as below for setting a new lot. When setting a lot newly, the action attribute in a <lot> tag is set at “create”.

Interface name		setLot	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	unit	Unit definition	0	No	3
	scale	Scale unit	0	No	2
	profile	Management information	0	1	1
	item	Information to be set (only ID)	0	No	1
○	lot	Information to be set	0	No	1
	order	Information to be set （only ID）	0	No	1

Interface name		setLot	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error information	0	No	1
	lot	Setting result （only ID）	0	No	1

```
<pslx type=“request” id=“a001” action=“setLot”>
<lot name=“X001” item=“product X” order=“K023” action=“create”>
<qty value=“130”/>
</lot>
</pslx>
```

When correcting the contents of the existing lot, the action attribute

value is set at “revise”. Setting the lot tracking information is regarded as correcting the lot information.

```
<pslx type=“request” id=“a001” action=“setLot”>
<lot name=“G003” action=“revise”>
<tracking lot=“G004”/>
</lot>
</pslx>
```

When deleting the specific lot information, the request message in which the action attribute of <lot> tag is ”delete” is sent as follows.

```
<pslx type=“request” id=“a001” action=“setLot”>
<lot name=“G003” action=“delete”/>
<lot name=“G004” action=“delete”/>
</pslx>
```

4.14.2. getLot

This interface inquires the lot information. The following messages are sent for getting the list of lot.

Interface name		getLot	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	profile	Management information	0	1	1
○	lot	Contents to be inquired	1	No	1

Interface name		getLot	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error contents	0	No	1
	unit	Unit definition	0	No	3
	scale	Scale unit	0	No	2
	operation	Inquiry result（lot tracking）	0	No	3
	lot	Inquriry result	0	No	1

```
<pslx type="request" id="a001" action="getList">
<lot name="#query"/>
</pslx>
```

The contents meeting the conditions are inquired. For example, when getting the quantity of lots of the specific item, the following message is sent.

```
<pslx type="request" id="a001" action="getLot">
<lot name="#query" item="product X"/>
</pslx>
```

Lot tracking can be executed. In such a case, sometimes the operation information is set as a result of inquiry besides the lot information.

4.15. Task Information

4.15.1. setTask

This interface sets the task information. The task information shows how much amount of the resources was used (or will be used) by the concrete indidivual operation in process. For setting the task information, the scheduler sends the result of scheduling to MES and so on, and conversely MES sets the information as the result to scheduler.

Interface name		setTask	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application extension	0	No	1
	unit	Unit definition	0	No	3
	scale	Scale unit	0	No	2
	profile	Management information	0	1	1
	item	Information to be set（only ID）	0	No	1

○	task	Information to be set	0	No	1
	order	Information to be set（only ID）	0	No	1

Interface name		setTask	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error information	0	No	1
	task	Setting result (only ID)	0	No	1

For example, the task information can be set with a request message as below.

```
<pslx type="request" id="a001" action="setTask">
<task name="Z001" resource="labor A" order="K023" action="create">
<qty value="15" unit="person・time"/>
</task>
</pslx>
```

For setting the task information, the information can newly be produced and also the contents can be corrected and deleted. In such a case, action attributes are ”revise” and ”delete” respectively.

In the response message to these request messages, the list of IDs of the task newly added, corrected or deleted is returned.

4.15.2. getTask

This interface inquires the task information. For inquiring the task information, the contents can be inquired individually with a task ID. However, usually the contents are inquired with setting the condition. The task on the specific resource or order is inquired or the contents is inquired with specifying the period.

Interface name		getTask	Division	Request	
Required	Element name	Explanation	min	max	Actual
	spec	For application	0	No	1

		extension			
	profile	Management information	0	1	1
○	task	Contents to be inquired	1	No	1

Interface name		getTask	Division	Response	
Required	Element name	Explanation	min	max	Actual
	error	Error contents	0	No	1
	unit	Unit definition	0	No	3
	scale	Scale unit	0	No	2
	task	Inquiry result	0	No	1

In the following example, the task contents on May 1 of resource A are inquired.

```
<pslx type="request" id="a001" action="getTask">
<task name="#query" resource="resource A">
<time><earliest value="2003-05-01:00:00:00"/>
<latest value=" 2003-05-02:00:00:00"/></time>
<query select="qty"/>
<query select="time"/>
</task>
</pslx>
```

The example of response message to the above inquiry is as below.

```
<pslx type="response" id="a001" ref="a001" action="getTask">
<task name="T0023" resource="resource A">
<time><value="2003-05-01:09:10:00"/></time><qty value="3"/>
</task>
<task name="T0034" resource="resource A">
<time><value="2003-05-01:14:30:00"/></time><qty value="5"/>
</task>
</pslx>
```

5. XML Data Exchange Specification (Part３)

This chapter shows XML Data Exchange Specification for implementing each interface of PSLX prescribed in Part 2 on individual application systems. In Part 3 explained in this chapter, the data contents that are diffrent in every interface aren't completely considered. Instead of that, the four-type messages to realize each interface is classified and the communication methods to those data are shown.

5.1. Placement of Data Exchange Specification

When two application programs execute data communication, the communication is managed by being divided into some layers as the following figure shows. In this specification, TCP/IP is used for the network layer, HTML for the transport layer and SOAP attachments for the middleware layer. This specification prescribes the part of a diagonal arrow in the figure that is the boundary between PSLX and SOAP with supposing the above structure.

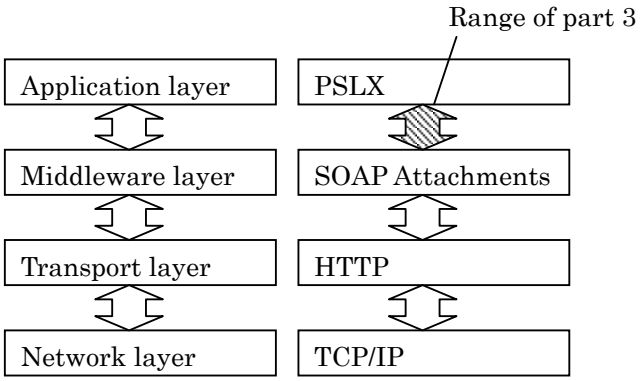
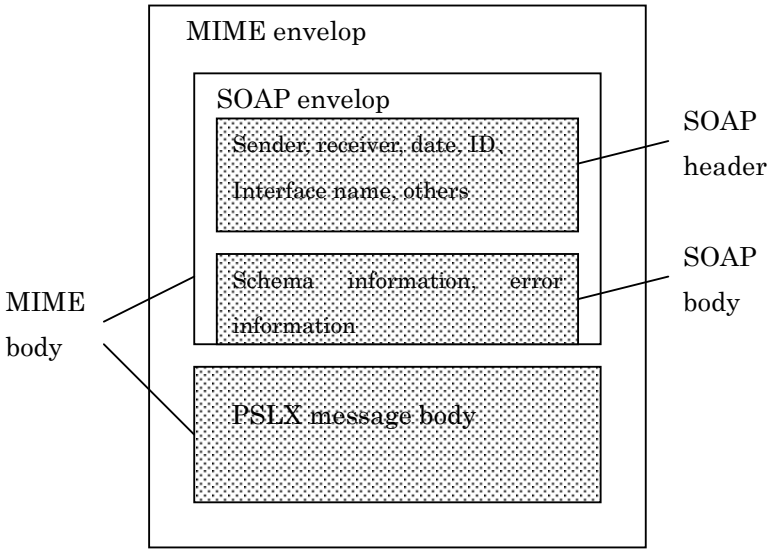


Figure Layer for Message Communication

The data described based on XML specification of PSLX is set in the

SOAP attachment using MIME envelope as the following figure shows. The information required for exchanging messages is set in the SOAP header in the envelope of SOAP and a manifest or the error information is set in the SOAP body.



Relation between PSLX and SOAP

5.2. Message Type

When PSLX data is exchanged between two application programs, there are four types of messages as below. However the data contents to make permanent data can be regarded as a message but it isn't included in the following classifications.

✧ **Request（request）message**

This is the message to request inquiries about any processing or information to the receiving application. When receiving a request message, a response message or an exception message must be returned to the sender. A receipt request can be added to the request message.

✧ **Response（response）message**

This is the message to respond to the application sending a request message by complying with the requested content. The response messages are the message to report a processing result and the message to return the contents of inquiry about data. A receipt request can be added to the response message.

✧ **Receipt（receipt）message**

This is the message to tell whether the receiving application accepts

the contents of message or not to the sending application in case that a receipt request is set in a request message or a response message. The application must send a receipt message or reject the contents by sending an exception message before executing each processing.

✧ **Exception（exception）message**

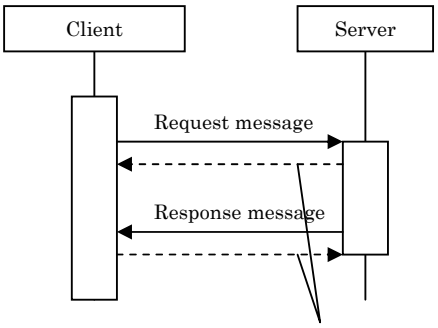
This is the message to notify the sending application that the application receiving a request message or a response message cannot continue processing because of the inconvenient contents. An exception message can be returned to a response message instead of a receipt message only when a receipt request is set. Even if there is a problem in the contents of receipt message or exception message, an exception message cannot be returned to them.

5.3. Basic Patterns

Data is exchanged between two applications by combining these messages. When not considering the existence of a receipt message, the following three patterns for exchanging messages are given according to the timing of exception occurrence. The side to send a request message is called a client and the responding side is called a server as below.

✧ **Request・response pattern**

This is the pattern that after a client sends a request message and a server receives the message, the applicable processing is executed and a result in the response message is returned with a response message. This pattern is most popular. The receiver sends a receipt message to the sender when a receipt request is set to a request message and a response message respectively.



Receipt message

Figure　Request・Response Pattern

✧ **Request・Exception Pattern**

This is the pattern that a server returns an exception message as an error to a client when the client sends a request message to the server but the server judges that there is an error in the message. The server may send a receipt message before sending an exception message when a receipt request is set in a request message.

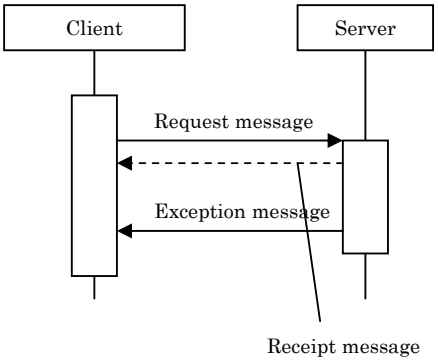


Figure　Request・Exception Pattern

✧ **Request・Response・Exception Pattern**

This is the pattern that a client sends an exception message to a server in case that the client knows that there is an error in the response message after the client sends a request message and the server returns a response message to the client. This pattern can be set only when a receipt confirmation request is sent with a response message. But a receipt message must not be returned when an exception message is returned to a response message.

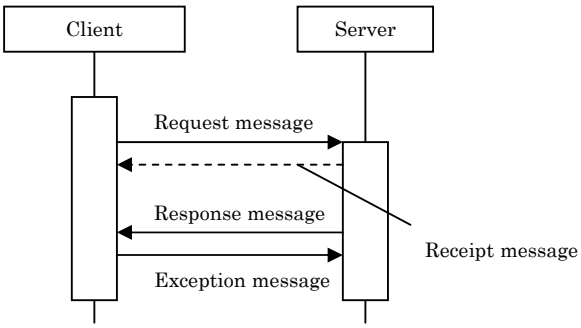


Figure　Request・Response・Exception Pattern

5.4. Message Exchange with HTTP

When communicating between applications with HTTP, a message always consists of a pair of request and response. Request messsge, response message, receipt message and exception message must be sent as a request or a response in any pattern in the following list.

Pattern	Request	Response
1	Request message	Response message
2	Request message	Exception message
3	Request message	No
4	Response message	No
5	Receipt message	No
6	Exception message	No
7	No	Request message
8	No	Response message
9	No	Receipt message
10	No	Exception message

"No" in the above list indicates that there is no data at the level of two application programs. In short, when there is no business data of PSLX expressed with XML in the SOAP attachment, the data is recognized as "No". However, there is the data on the communication level of HTTP or SOAP.

A server and a client are divided into the application level to request and respond, and the sender and receiver level that actually communicates, and then each communication pattern is explained. Only the data exchange methods between a sender and a receiver are the convention. The flow of activities other than them and the relation among sender, receiver and application are implementation samples.

✧ **Pattern 1 , Pattern 2**

Firstly, in pattern 1 and pattern 2, a response message or an exception message is returned to a request message. The flow of processing is the form like the following figure. To the message from a sender, an error communication may be returned (HTTP error); an error may be returned from a receiver (HTTP error); a response message or an exception message may be returned from a responder.

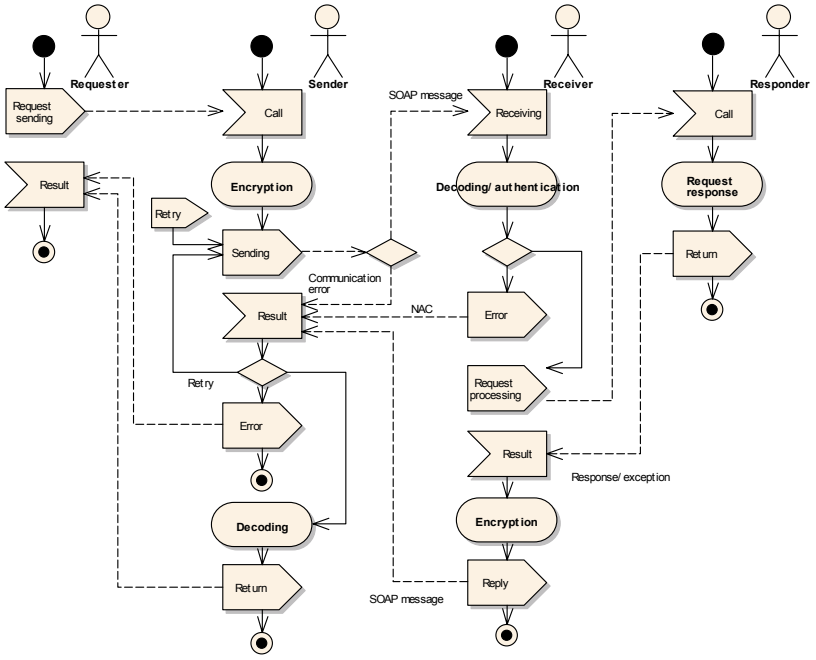


Figure　Sending Request Message（Pattern 1 , 2）

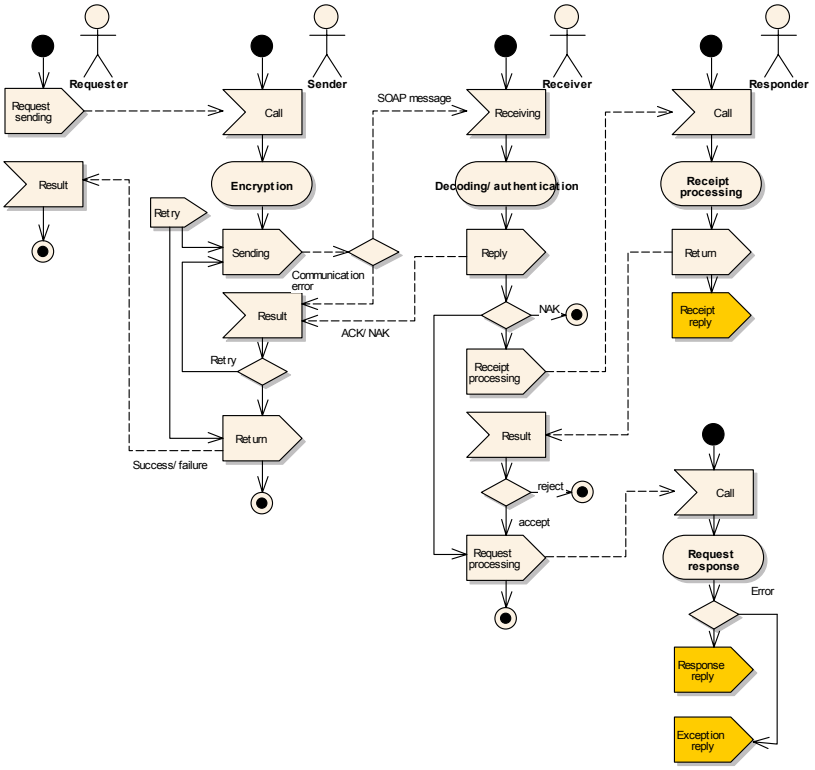


Figure　Sending Request Message（Pattern3）

✧ **Pattern 3**

In pattern 3, a request message is sent but a receipt message, a response message and an exception message to the request message are handled in another session. The communication form is as the following figure shows. To the request message from a sender, a communication error may be returned (HTTP error); an error may be returned from a receiver (NAC : HTTP error); the message may be received normally（ACK : HTTP normal response）.

✧ **Pattern 4**

In pattern 4, a response message to a request message is returned to a requester in the form as below. A receipt message and an error message aren't be returned to a response message together. Therefore there are three cases to a response message; a receipt message is sent; an exception message is sent; nothing is returned. When sending nothing, it doesn't mean that there is always no error in a response

message.

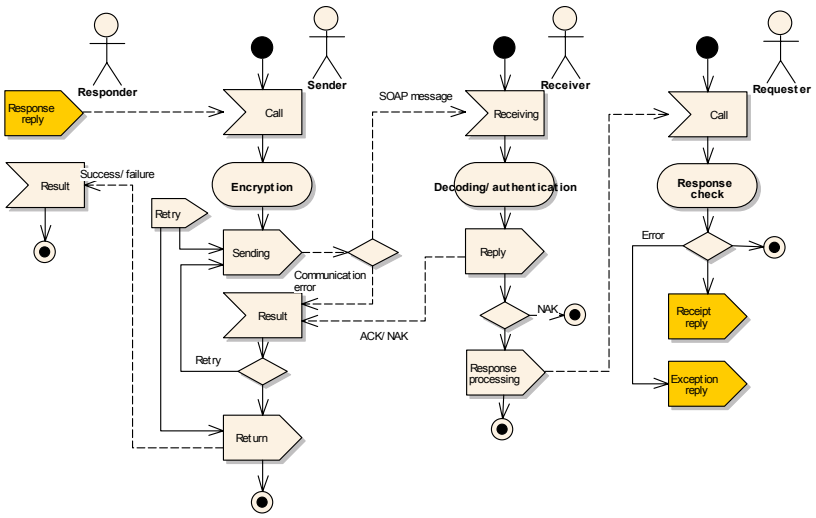


Figure　Sending Response Message（Pattern4）

✧ **Pattern 5**

In pattern 5, the following communication method is used to return a receipt message to a sender. A receipt message may be created for both request message and response message. The receiver of a receipt message informs a responder or requester who is the address of a receipt message that the message is received. A newly created message isn't sent back to a receipt message.

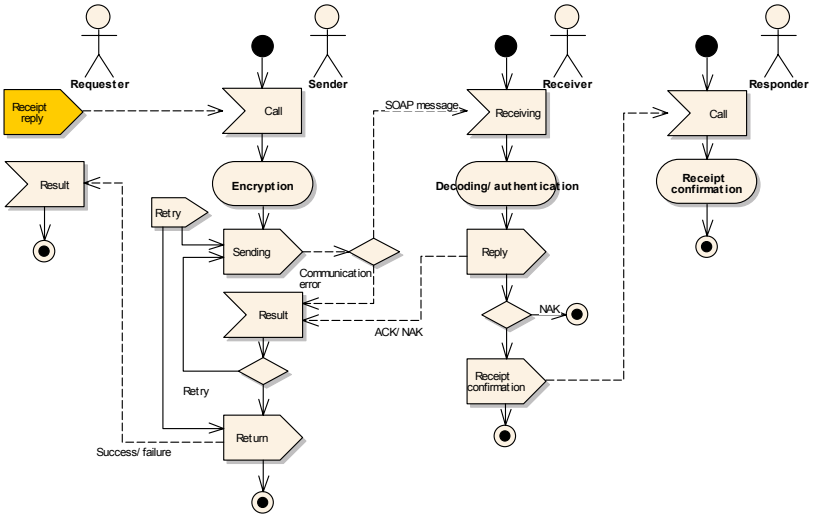


Figure　Sending Receipt Message（Pattern 5）

✧ **Pattern 6**

In pattern 6, an exception message is returned to a sender. Sometimes an exception message is returned to a request message and a response message respectively. The exception message is sent in the form as below. A newly created message isn't returned to the exception message. Even if there is an error in the exception message, the error cannot be notified to the sender. However an error in communication or an error judged by the receiver such as authentication can be returned as a HTTP error.

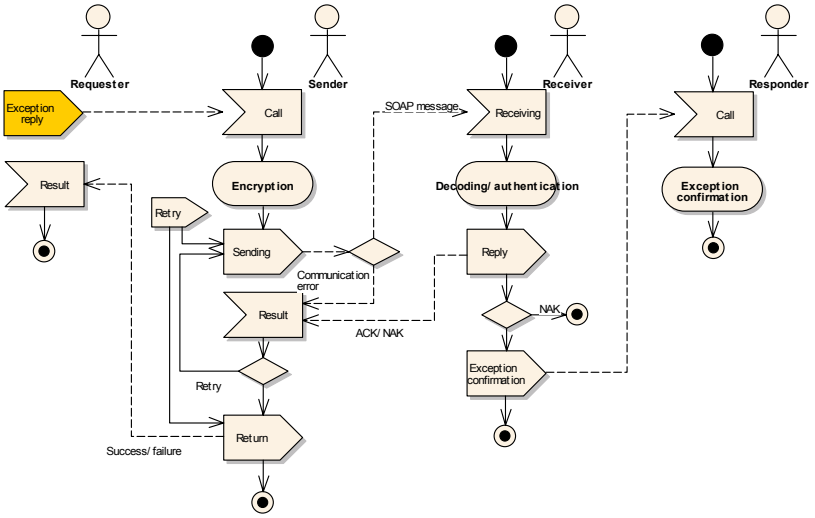


Figure　Sending Exception Message（Pattern 6）

✧ **Patterns 7～１０**

Finally, patterns 7 to 10 are the pull-type communication method. The following pull-type communication method is applicable for request message, response message, receipt message and exception message.

The feature of this pattern is that a sender of message is a responder of HTTP.　A requester registers a request in the message queue of the sender beforehand and waits for the inquiry from a receiver.　The receiver accesses to the registered sender at regular intervals and if there is a message to the receiver, it is taken in.

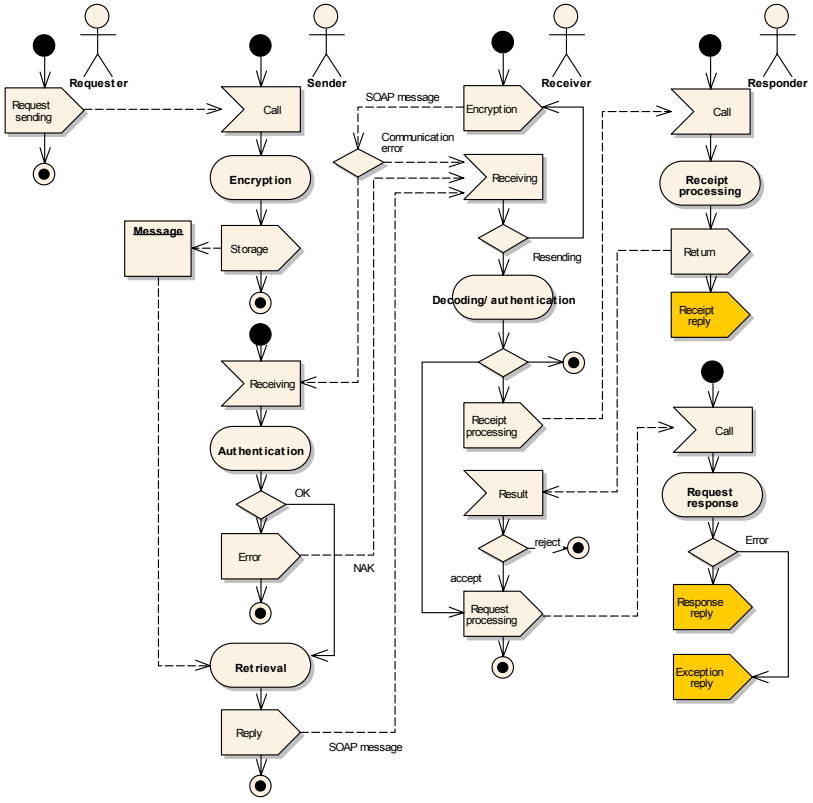


Figure　Pull-type Message Sending (Pattern7-10)

5.5. Synchronous Communication and Asynchronous Communication

PSLX supports both synchronous communication method and asynchronous communication method. There are Push-Push type, Push-Pull type and Pull-Push type in asynchronous type. Each communication site must declare the applicable communication method out of these communication methods. In case of asynchronous type, if a receipt request cannot be set to a request message or a response message, it must be declared beforehand.

✧ **Synchronous communication**

When executing synchronous communication, a request message must correspond to Request and a response message or an exception message to a request message must correspond to Response as pattern 1 and pattern 2 show.

In case of synchronous communication, a receipt request cannot be added. Therefore a receipt message isn't set in both Request and Response. Even if there is an error in a response message, the error cannot be notified to the server.

The following shows message patterns in case of synchronous communication. The process is always completed with one time of go-and-return in synchronous communication.

C→S・S→C	C→S・S→C	C→S・S→C	C→S・S→C
Request・Response			
Request・Exception			

✧ **Asynchronous Push-Push communiction**

Asynchronous communication can be divided into interactive type (Push-Push), push type (Push-Pull) and pull type (Pull-Push). The type depends on which a server or a client sends a request message, a response message and so on with HTTP.

For instance, when considering the case where a request message is sent from a client to a server, there are two cases; a client is a sender of HTTP and sends a message to a server as Request（Push）; a server is a sender of HTTP and takes out a message from a client by Response（Pull）.

In asynchronous Push-Push type, a message is always sent as a request of HTTP as below.

●Request・Response pattern

C→S・S→C	S→C・C→S	S→C・C→S	C→S・S→C
Request・－		Response・－	
Request・－		Reponse・－	Receipt・－
Request・－	Receipt・－	Response・－	
Request・－	Receipt・－	Response・－	Receipt・－

●Request・Exception pattern

C→S・S→C	S→C・C→S	S→C・C→S	C→S・S→C
Request・－		Exception・－	
Request・－	Receipt・－	Exception・－	

●Request・Response・Exception pattern

C→S・S→C	S→C・C→S	S→C・C→S	C→S・S→C
Request・－		Response・－	Exception・－
Request・－	Receipt・－	Response・－	Exception・－

✧ **Asynchronous Push-Pull communication**

Asynchronous Push-Pull type is the communication method that a client is always a sender of HTTP. Therefore a request message is sent in Push（request）and a response message is returned in Pull（response）.

In this method, a client needs to access to a server regularly and confirm the message in order to get a response message, exception message and receipt message from the server.

●Request・Response pattern

C→S・S→C	C→S・S→C	C→S・S→C	C→S・S→C
Request・－		－　・Response	
Request・－		－　・Response	Receipt・－
Request・－	－　・Receipt	－　・Response	
Request・－	－　・Receipt	－　・Response	Receipt・－

●Request・Exception pattern

C→S・S→C	C→S・S→C	C→S・S→C	C→S・S→C
Request・－		－　・Exception	
Request・－	－　・Receipt	－　・Exception	

●Request・Response・Exception pattern

C→S・S→C	C→S・S→C	C→S・S→C	C→S・S→C
Request・－		－　・Response	Exception・－
Request・－	－　・Receipt	－　・Response	Exception・－

✧ **Asynchronous Pull-Push communication**

In asynchronous Pull-Push type communication, a server is a sender of HTTP and takes out a request message from a client. In short, the request message is a response of HTTP.

For communicating in this method, a sever needs to confirm whether each of registered clients has a request or not at regular intervals. A receipt message and an exception message to a response message are confirmed similarly.

●Request・Response pattern

S→C・C→S	S→C・C→S	S→C・C→S	S→C・C→S
－　・Request		Response・－	
－　・Request		Response・－	－　・Receipt

●Request・Exception pattern

S→C・C→S	S→C・C→S	S→C・C→S	S→C・C→S
－　・Request		Exception・－	
－　・Request	Receipt・－	Exception・－	

●Request・Response・Exception pattern

S→C・C→S	S→C・C→S	S→C・C→S	S→C・C→S
－　・Request		Response・－	－・Exception
－　・Request	Receipt・－	Response・－	－・Exception

5.6. How to Procee Error

The application program that is a server or a client issues an exception message indicating error occurrence. On the other hand, a receiver

also creates and returns an error message to a sender at the communication program level. Moreover there are errors that cannot arrive at a communication program of the party and are created at the lower level such as the error occurred by a physical problem.

When classifying these errors, there are the following levels. Error code in PSLX is put in parentheses. This code correspond to the error code set in an exception message.

（１） Communication error on the network（１００１～）

（２） Error in SOAP syntax or meaning（２００１～）

（３） Error in authentication by communication program and so on（３００１～）

（４） Error between a communication program and an application（４００１～）

（５） XML schema syntax error in a PSLX message（５００１～）

（６） Verification error of the contents model by PSLX specific constraint（６００１～）

（７） Error found only by executing processing logic（７００１～）

When the business application of the receiver sets an error, the error has the exception message form from the beginning. The message code doesn't correspond to the error that occurs in a communication program or at lower level. In such a case, it is recommended that the error contents are converted to a PSLX error code in the communication program in order that the final application on the side to receive the error can recognize the error contents as occasion demands.

The character to distinguish error and warning is set in the end of error code as below.

Identification character	Explanation	Example
E	Show an error	7002E

W	Show warning	6001W

✧ **Agreement on resending**

The application receiving a request message must return a response message or en exception message without fail. Therefore if the message isn't returned, the sent request message may not arrive at the application program of the receiver or the receiving application may not return a message yet or the message returned by the receiving application may not arrive at the sender.

In such a case, the application sending a request message can send the same message to the receiver within the specific number of times at regular intervals.

If a receipt request is added to a response message, the receiving application must surely return a receipt message or an exception message. Therefore if any message isn't returned, a response message may not reach or the application receiving a response message may not send a reply yet, or the returned message may not arrive at the sender. When a response message may not be returned yet, a response message may not deal with a receipt request.

In such a case, the application sending a response message can send the same message to the receiver within the specific number of times at regular intervals.

When resending a message, the message ID must be the same in order to show that the message is the same. This message ID must be a unique character string in the sending application.

The application receiving some request messages with the same message ID can ignore the second and later messages if the application doesn't return a response message yet.

If the same message is sent to the application after a response message or an exception message is returned to a request message, a response message or an exception message must be resent adding the same ID as the ID sent before. However, when the number of repeat times is over the fixed range, the message need not be sent any more.

If the same response message is sent after a receipt message or an exception message is returned to a response message, a receipt message or an exception message must be resent within the specific number of times. If the number of repeat times is over the specific number, the message must not be sent any more.

5.7. SOAP Usage and ebXML

In PSLX, XML data expressed with the tag structure prescribed in part 1 is sent after the data is packaged using SOAP 1.1 Attachments. PSLX data is set as the appendix information of SOAP as a MIME part. The way of using the header and body of SOAP has the structure based on ebXML messaging service convention.

Therefore the schema information and a manifest are specified in the body. Interface name（action name）, service name（application name）, the receiver information and the sender information are set in SOAP header. The following shows tag names and the setting information in SOAP.

Tag structure	Explanation	min	max
Header		1	1
MessageHeader		1	1
From	Information about sender	1	1
PartyID		1	No
To	Information about receiver	1	1
PartyID		1	No
Service	Application information	1	1
Action	Interface identification information	1	1
MessageData	Message identification information	1	1
MessageID		1	1

The data in which URI of the site is set must be set in PartyID subelement of From element and PartyID subelement of To element respectively after setting type="uri". The URI of the server providing

Web service must be set here.

The character string to identify an application program must be set in Service element.

The interface name supported by the application program specified with Service must be specified in Action attribute. The interface name in this section corresponds to the interface name in part 2 in this specification with ”Request”, ”Response”, ”Exception” and ”Receipt”. For example, when sending the interface, setSchedule as a request message, setScheduleRequest is set.

MessageID must be the combination of PartyID and Service code of the application creating a message and the ID to be set in a PSLX message and must be a completely unique character string.

5.8. Error Code

	Error in a communication program	The error that cannot be recovered occurs in a communication program. The error contents received from the communication program is shown as well.
	XML syntax error	The message contents have a syntax error in XML.
	PSLX schema verification error	The message violates the syntax independently set by PSLX such as the tag structure
	PSLX schema verification error（XXX）	The message violates the schema set in every interface in PSLX specification.
	Interface implementation error	The applicable interface isn’t implemented.
	Interface name identification error	Interface name cannot be identified.

	Message ID error	The messag ID isn't a unique value.
	Date setting error	There is an error in the precedence of date such as start and end.
	Unit specification error	The unit specified with unit attribute cannot be identified.

1

Appendix A　Relation with Domain Object

The XML tag structure prescribed in this specification is the structure based on Part 3 “PSLX Domain Objects” in PSLX Engineering Specification. The domain object shown in Part 3 is converted to the XML schema structure in this specification following the policy as below. The terms of domain object are expressed with「」 and elements of XML tag are expressed with “” for explanation.

（1）Integrating the master information and the instance information

The master information and the instance information are expressed in the same class without classifying them. For example, the master information,「operation」and「schedule」 are “operation (operation)”. Both「event」and「occurrence」are “event（event)”. 「item」and「substance」are “item（item)”.

（2）Dividing a class or abolishing a parent class

The class that is 「item」 but not 「resource」is regarded as “item (item)” and 「resource」is regarded as “resource (resource)” to 「item」 and its subclass「resource」as a basic class. With this,「task」that is a subclass of「lot」is regarded as the same class.

「produce」 class is divided into “production（produce）”, “consumption(consume)”and “assignment(assign)”, and「interval」 is divided into “interval（interval)” and “switch（changeover)”. 「party」 is abolished. 「customer」 and 「supplier」are the highest classes and are taken as “customer（customer)” and “supplier (supplier)”.

（3）Simplifying the class relation structure by deleting the relation

Under 「rule」,「interval」,「mode」,「produce」,「action」 and 「sd」 relate to「operation」but only「produce」is regarded as being related with 「operation」. “Produce”, “consume” and “assign” are restricted within the range where they can be expressed with the combination of nth attribute and sel attribute.

（４）Integrating the similar classes related with plan

「calculation」, 「subject」,「plan」and「sd」indicating the numerical information or the calculation method are integrated into “parameter（parameter）”.

（５）Integrating and sub-classifying the information related with feature

「feature」 of various 「item」and the value, 「state」are integrated into “specification（spec）”. “Stock (stock)”, “load (load)”, “capacity (capacity)” and “location (location)” , which are used frequently as a subclass of “specification”, are set separately.

（６）Simplifying the definition by the pattern information

The information about “calendar (calendar)” , which is not an essential element but is convenient, is defined with “shift（shft）” information. It can be said that “shift” is the pattern data of 「capacity」class. “Shift” and “calendar” can be rewritten completely by 「capacity」information.

（７）Classifying the common attribute information

There are many data that should essentially be taken as an attribute because of operating XML tag. For instance, the attribute data such as 「time」, 「quantity」 and 「price」 become classes , “time（time）”, ”quantity（qty）”and “unit price（price）” respectively. And also 「lotsize」and「tasksize」are the class information.

（８）Shifting the class information by tracing the relation

「Precedence」isn’t directly defined in 「operation」and so the information can be traced through operation events. But in this section, operation is defined as “predecessor（predecessor）” and “successor（successor）” of operation. For instance, this section sets the premise that the relation between “start event” of the operation and “end event” of predecessor corresponds to 「precedence」in “predecessor”.

（９）Classifying the related information

The relation indicating 「partof」is the class, “inclusion (partof)” as the relation between classes. “Partof” is set in four classes, “item”, “resource”, “operation” and “order”. So the numerical value of partof can be set newly.

PSLX domain object can be converted to XML tag structure by operating as above, and adding, correcting and deleting the attribute and the relation of each class. The following shows the relation between the tag element in XML schema and the PSLX domain object.

Class name		English	Tag name
Item		item	Item
	Product	product	Item
	Material	material	Item
	Work in process	wip	Item
	Middle product	subassy	Item
Article		substance	Item
Resource		resource	Resource
	Equipment	equipment	resource
	Tool	tool	resource
	Labor	labor	resource
	Workshop	ws	resource
	Working section	shop	resource
	Management section	site	resource
Feature		feature	spec
	Stock	stock	stock
	Load	load	load
	Capacity	capacity	capacity
	Location	location	location
State		state	spec
	Amount of stock	stock value	stock

	Amount of load	load value	load
	Amount of capacity	capacity value	capacity
	Location contents	location value	location
Calculating method		subject	parameter
Calculation value		calculation	parameter
	Cost	cost	parameter
	Profit	profit	parameter
Management unit		sd	parameter
Plan		plan	parameter
	Stock plan	parameter	parameter
	Load plan	load plan	parameter
	Capacity plan	capacity plan	parameter
	Sales plan	sales plan	parameter
	Purchase plan	purchase plan	parameter
Constraint		constraint	expression
Event		event	event
	Start	start	start
	End	end	end
	Suspend	suspend	event
	Resume	resume	event
Occurrence		occurrence	event
Operation		operation	operation
	Fabrication	fabrication	operation
	Transportation	transportation	operation
	Storage	storage	operation
	Inspection	inspection	operation
	Setup	setup	operation
	Maintenance	maintenance	operation
	Design	design	operation
	Production	production	operation
	Sales	sales	operation

	Purchase	purchase	operation
Operation attribute		mode	spec
Operation schedule		schedule	operation
Operation progress		progress	operation
Time relation		precedence	predecessor, successor
Operation relation		interval	interval
Switch relation		changeover	changeover
Action		action	action
Condition		condition	condition
Production		produce	produce
	Consumption	consume	consume
	Assignment	assign	assign
Lot		lot	lot
	Task	task	task
Order		order	order
	Product order	product order	order
	Forecast order	forecast order	order
	Customer order	customer order	order
	Process order	process order	order
	Outsourcing order	outsourcing	order
	Purchase order	purchase order	order
	Transportation order	transportation order	order
Party		party	customer
	Customer	customer	customer
	Supplier	supplier	supplier
Production rule		rule	produme,consume,assign
Pegging		pegging	pegging
Tracking		tracking	tracking

1