API & Schema Reference Manual

BillerXpert 4.6

April 2002

Contents

List of Tables

About This Document

The BillerXpert *4.6 API & Schema Reference Manual* describes the interfaces to component business objects that you can use to customize BillerXpert 4.6. It also describes the database schemas for BillerXpert.

This preface contains the following sections:

- Audience
- What's in This Document
- Documentation Conventions
- The Document Online
- Related Documentation
- Product Support

Audience

The audience for this document is web site developers and Java programmers who want to modify the appearance of BillerXpert or its operation. You must have experience with HTML and Java and have access to the iPlanet Application Server (iAS) documentation:

```
http://docs.iplanet.com/docs/manuals/ias.html
```

You need to use this manual if you are changing or extending BillerXpert, changing the database schema, or adding event handlers.

Required Documents

When customizing BillerXpert, you may need to refer to the *iPlanet BillerXpert 4.6 Customization Guide* for information about and examples of using component business object interfaces and event handlers.

You will also need to have read or have available the following iPlanet Application Server (iAS) documents:

- i*AS Java Programmer's Guide* — Provides background information about using the iAS programming constructs upon which BillerXpert was developed. (`http://docs.iplanet.com/docs/manuals/ias/60/sp2/JavaProgGuide/contents.htm`.)
- *IAS Foundation Class Reference (Java)* — Provides descriptions of classes that BillerXpert uses or classes from which BillerXpert classes are derived. (`http://docs.iplanet.com/docs/manuals/ias/60/sp2/javaapi/contents.htm`.)

What's in This Document

The following table summarizes what each chapter covers.

Table 1 Chapter Summary

If you want to know about this	See this chapter
Describes the contents of this guide; listing of documentation set; information on product support.	"About This Document"
Introduction to the component business object interface classes and database schemas for BillerXpert	"Introduction to BillerXpert 4.6 Classes and Database Schemas"
Lists the following BuyerXpert Java classes and methods: • Base and core classes • Billing • Membership • Event handlers	Part 1, "BillerXpert 4.6 Classes"
Describes the interfaces to the base and core classes, which provide basic functions such as activity tracking or error handling, which is shared by all BillerXpert components.	Chapter 1, "Base and Component Business Object Interfaces"
Describes the component business object interface classes for billing records. These classes provide access to records stored in an Oracle database.	Chapter 2, "Billing Classes Component Business Object Interfaces"

Table 1 Chapter Summary

If you want to know about this	See this chapter
Describes the component business object interface classes for membership records. These classes provide access to records stored in a Directory Server database using the Lightweight Directory Access Protocol (LDAP).	Chapter 3, “Membership Component Business Object Interfaces”
Describes the component business object interface classes for events. These classes provide events and event handlers for handling asynchronous events within BillerXpert.	Chapter 4, “Event Component Business Object Interfaces
Lists the following BuyerXpert schemas: • Billing schema • Membership schema	Part 2, “BillerXpert Database Schemas
Describes the structure of the Oracle database used by BillerXpert	Chapter 5, “Billing Schema
Describes the structure of the Directory Server database used by BillerXpert.	Chapter 6, “Membership Schema
Describes the domains that an implementation of a component business object interface checks before performing actions against a data repository.	Appendix A, “Enumerated Domains

Documentation Conventions

This document uses the following conventions:

- The `monospace` font is used for sample code and code listings, Application Program Interface (API) and language elements (such as method names and property names), file names, path names, directory names, Hypertext Markup Language (HTML) tags, and any text that must be typed on the screen.
- The *italic* font is used in code to represent placeholder parameters (variables) that should be replaced with an actual value.
- Brackets ([]) are used to enclose optional parameters.
- A slash (/) is used to separate directories in a path. (Windows NT supports both the slash and the backslash.)

The Document Online

For an overview of BillerXpert 4.6 changes, refer to the BillerXpert 4.6 Release Notes.

An electronic version of these notes are available at:

`http://docs.iplanet.com/docs/manuals/`

Related Documentation

The BillerXpert documentation set includes:

- *BillerXpert 4.6 Release Notes*—Contains important information on the current release of BillerXpert. Read this document before working with the new BillerXpert release.
- *BillerXpert 4.6 Installation Guide*—Provides instructions for installing the BillerXpert product and its enabling software.
- *BillerXpert 4.6 Administrator's Guide*—Provides reference information and instructions on administering a fully-installed BillerXpert system.
- *BillerXpert 4.6 BSP Administration Guide*—Provides reference information and instructions on administering the BSP functionality of a fully-installed BillerXpert system.
- *BillerXpert 4.6 Company Administration Guide*—Provides reference information and instructions on administering the company functionality of a a fully-installed BillerXpert system.
- *BillerXpert 4.6 Customization Guide*—Provides guidelines and instructions for customizing the BillerXpert system.
- *BillerXpert 4.6 Integration Guide*—Provides guidelines and instructions for integration the BillerXpert system.

Documentation for all iPlanet products can be found at the following web site:

`http://docs.iplanet.com/docs/manuals/`

Product Support

If you have problems with your BillerXpert system, contact iPlanet customer support using one of the following mechanisms:

- iPlanet online support web site at:

  `http://www.iplanet.com/support/online/`

  From this location, the CaseTracker and CaseView tools are available for logging problems.

- The telephone dispatch number associated with your maintenance contract

So that the technical support staff can best assist you in resolving problems, please have the following information available when you contact support:

- Description of the problem, including the situation where the problem occurs and its impact on your operation
- Machine type, operating system version, and product version, including any patches and other software that might be affecting the problem
- Detailed steps on the methods you have used to reproduce the problem
- Any error logs or core dumps

Introduction to BillerXpert 4.6 Classes and Database Schemas

This chapter introduces the component business object interface classes and database schemas. For information about BillerXpert application-related classes, see the *iPlanet BillerXpert 4.6 Customization Guide*.

This chapter presents the following topics:

- Component Business Object Interface Classes
- Database Tables

Component Business Object Interface Classes

BillerXpert contains component business objects that are Java interface based. There are two major components:

- Membership
- Billing

These component objects manage data storage, such as creation, storage, and retrieval of records, or contain processing logic.

Component Objects

Each component has its own factory class, which is used to create the object, and its own context class. A component object must be associated with a context before it can perform data access or processing. The responsibilities of these contexts differ depending on the kind of component.

Table 2 lists these components:

Table 2 Component Objects

Component	Context Responsibilities
Membership	None.; however, the context is required by convention.
Billing	Establishes a context between the component and the component object; responsible for database connections and transaction management.

Base and Core Classes

Component business object classes are derived from the IBXEntity class. In addition, BillerXpert provides several core classes that you can use with component business objects.

Table 3 shows the base classes for component business object interfaces and their contexts. These classes are located in Chapter 1, “Base and Component Business Object Interfaces.

Table 3 Base and Core Classes *(1 of 3)*

Class Name	Description
1. ACFormatMessage	Represents message format information
2. ACModules	Represents the list of BX modules
3. ACModuleStruct	Represents the structure of the BX modules
4. ACStatus	Represents AC Status
5. BaseAppLogic	Represents base core class
6. BXAdminActivityALDef	Represents the Access list definition for the Admin
7. BXAdminBaseAppLogic	Represents the base core applogic for the Admin
8. BXAdminBaseView	Represents the base applogic view for the Admin
9. BXAdminConfigObj	Represents the configuration object for the Admin
10. BXAdminDateUtil	Represents the date utility object for the Admin
11. BXAdminErrorCode	Represents the error code container for the Admin
12. BXAdminEventDBDef	Represents the event database definition for the Admin
13. BXAdminLoadingALDef	Represents the Loading Access list for the Admin
14. BXAdminLoadingDBDef	Represents the Loading database definition for the Admin

Table 3 Base and Core Classes *(2 of 3)*

Class Name	Description
15. BXAdminMemALDef	Represents the Admin membership Access list
16. BXAdminMemDBDef	Represents the membership database definition for the Admin
17. BXAdminMessages	Represents the Admin message module
18. BXAdminObjConfigALDef	Represents the Admin object configuration
19. BXAdminObjConfigDBDef	Represents the Admin database object configuration
20. BXAdminPaymentDBDef	Represents the Admin payment database object
21. BXAdminSession	Represents the Admin session session
22. BXAdminStaticResources	Represents the Admin resources
23. BXAdminTemplateDataBasic	Represents the Admin templates
24. BXAdminTemplateMapBasic	Represents the Admin template mapping
25. BXAdminUtil	Represents the Admin Utility
26. BXAdminValidateAttribute	Represents the Admin validation attribute
27. BXAdminValidateView	Represents the Admin validation view
28. BXAdminValidator	Represents the Admin validator
29. BXBaseAppLogic	Represents the Base Applogic
30. BXBaseDef	Represents the Base Definition
31. BXConfig	Represents configuration information.
32. BXConfigObj	Represents configuration object
33. BXConverter	Represents converter object
34. BXDatetime	Represents date information
35. BXDBContext	Represents the database context object
36. BXDBDef	Represents the database object definition
37. BXDef	Represents the definition object
38. BXEntityObject	Represents the Entity Object
39. BXEnumDef	Represents the Enumarated Domain object
40. BXError	Represents Error information
41. BXException	Represents exception information
42. BXFileId	Represents FileID information
43. BXFilter	Represents a search filter.

Table 3 Base and Core Classes *(3 of 3)*

Class Name	Description
44. BXGlobal	Represents global information
45. BXJDBCHelper	Represents the JDBC helper class information
46. BXLdapConfig	Represents LDAP config class
47. BXMessages	Represents message class information
48. BXObjectMgr	Represents object manager class information
49. BXObjMap	Represents object map class information
50. BXOrder	Represents the order in which search results are returned.
51. BXStatus	Represents a stack of status information.
52. BXStopwatch	Represents configuration information
53. BXTableDef	Represents Table definition class information
54. SearchResultSet	Represents search / result class information
55. BXValidator	Represents validator class information

Billing Classes

Billing classes represent billing records in the database. You use these classes to manipulate billing objects.

Table 4 shows the billing classes and interfaces for component business object interfaces and their contexts. These classes are located in Chapter 2, “Billing Classes Component Business Object Interfaces.

Table 4 Billing Classes *(1 of 2)*

Class Name	Description
1. BSPBillerInfo	Base class that manipulates BillerXpert information objects.
2. IBSPBillerInfo	Interface that manipulates BillerXpert information objects.
3. BXBillerCustAcct	Base class that manipulates BillerXpert customer account objects.
4. IBXBillerCustAcct	Interface that manipulates BillerXpert customer account objects.
5. BXBillingContext	Communicate between an AppLogic and an EJB component.
6. BXBillingEntity	Base class for billing business objects.
7. IBXBillingEntity	Interface for billing business objects.
8. BXBillingMgr	Creates billing-related component business objects.

Table 4 Billing Classes *(2 of 2)*

Class Name	Description
9. BXCardInfo	Base class for credit card information.
10. IBXCardInfo	Interface for credit card information.
11. BXClickThrough	Base class that manipulates click through objects.
12. IBXClickThrough	Interface that manipulates click through objects.
13. BXDetailCall	Base class that manipulates detail call objects.
14. IBXDetailCall	Interface that manipulates detail call objects.
15. BXDetailMisc	Base class that manipulates miscellaneous detail objects.
16. IBXDetailMisc	Interface that manipulates miscellaneous detail objects.
17. BXDetailRating	Base class that manipulates detail rating objects.
18. IBXDetailRating	Interface that manipulates detail rating objects.
19. BXDocument	Base class that manipulates billing objects.
20. IBXDocument	Interface that manipulates billing objects.
21. BXOFXSignOn	Base class that manipulates OFX sign on objects.
22. IBXOFXSignOn	Interface that manipulates OFX sign on objects.
23. BXPayHistory	Manipulates payment history objects.
24. IBXPayHistory	Manipulates payment schedule objects.
25. BXPaySchedule	Base class that manipulates payment schedule objects.
26. IBXPaySchedule	Interface that manipulates payment schedule objects.
27. BXSummary	Base class that manipulates billing summary objects.
28. IBXSummary	Interface that manipulates billing summary objects.
29. CSXBXMap	Base class that manipulates CSX map objects.
30. ICSXBXMap	Interface that manipulates CSX map objects.
31. OFXRequestLog	Base class that manipulates OFX request log objects.
32. IOFXRequestLog	Interface that manipulates OFX request log objects.

Membership Classes

Membership classes represent LDAP records on the Directory Server. You use these classes to manipulate membership objects.

Table 5 shows the membership classes and interfaces for component business object interfaces and their contexts. These classes are located in Chapter 3, “Membership Component Business Object Interfaces.

Table 5 Membership Classes / Interfaces

	Class Name	Description
1.	BXAccountType	Base class that manipulates account type objects.
2.	IBXAccountType	Interface that manipulates account type objects.
3.	BXAddress	Base class that manipulates address objects.
4.	IBXAddress	Interface that manipulates address objects.
5.	BXBillerGroup	Base class that manipulates biller group objects.
6.	IBXBillerGroup	Interface that manipulates biller group objects.
7.	BXBillerProfile	Base class that manipulates biller profile objects.
8.	IBXBillerProfile	Interface that manipulates biller profile objects.
9.	BXCustomerProfile	Base class that manipulates customer profile objects.
10.	IBXCustomerProfile	Interface that manipulates customer profile objects.
11.	BXFinancialAccount	Base class that manipulates financial account objects.
12.	IBXFinancialAccount	Interface that manipulates financial account objects.
13.	BXGroup	Base class that manipulates group objects.
14.	IBXGroup	Interface that manipulates group objects.
15.	BXMembershipContext	Establishes a context between an AppLogic and a membership business object.
16.	BXMembershipEntity	Base class that manipulates membership entity objects.
17.	IBXMembershipEntity	Interface that manipulates membership entity objects.
18.	BXMembershipMgr	Creates membership-related business objects.
19.	BXPaymentType	Base class that manipulates payment type objects.
20.	IBXPaymentType	Interface that defines payment type objects.

Event Classes

Event classes represent event records in the database. The event server uses these classes to manipulate event-related objects.

Table 6 shows the event classes and interfaces for component business object interfaces and their contexts. These classes are located in Chapter 4, “Event Component Business Object Interfaces.

Table 6 Event Classes

Class Name	Description
1. BXEvent	Base class that manipulates event objects.
2. IBXEvent	Base class that manipulates event-related objects.
3. BXEventContext	Base class that manipulates event context objects.
4. BXEventEntity	Base class that manipulates event-related objects.
5. IBXEventEntity	Interface class that manipulates event-related objects.
6. BXEventHandler	Base class that manipulates event handler mapping objects.
7. IBXEventHandler	Interface class that manipulates event handler mapping objects.
8. BXEventMgr	Manipulates event manager objects.
9. BXEventQueue	Base class that manipulates event queue objects.
10. IBXEventQueue	Interface class that manipulates event queue objects.
11. BXHandler	Base class that manipulates handler objects.
12. IBXHandler	Interface class that manipulates handler objects.

Database Tables

BillerXpert contains the following database tables:

- Billing Schema
- Member Schema

Table 7 identifies the tables in the Oracle database for BillerXpert.

Table 7 Database Tables

Schema	Description
1. BXActivity	Activity Log
2. BXBillerCustomerAccount	BillerXpert customer account
3. BXContainer	Containers
4. BXDocLineAsgt	Relationship between documents and detail lines
5. BXDocument	Documents

Table 7 Database Tables

Schema	Description
6. BXError	Error Logs
7. BXEvent	Events
8. BXEventHandler	Event handlers
9. BXEventLog	Event logs
10. BXEventLogDetail	Event log detail lines
11. BXEventQueue	Event queue entries
12. BXEventSchedule	Event Scheduling
13. BXExtensionDesc	Extension Description
14. BXHandler	Handlers
15. BXLineItemTelCoAll	Call details
16. BXLineItemTelCoMisc	Miscellaneous details
17. BXLineItemTelCoRating	Rating details
18. BXLoadTrack	Loader tracking
19. BXMessage	Messages
20. BXPayHistory	Payment histories
21. BXPaySchedule	Payment schedules
22. BXSummary	Summary lines
23. BXUserAcctAsgt	Relationship between users and accounts
24. BXCardInfo	Stores credit card related information

LDAP Schema

Table 8 identifies the LDAP schema for BillerXpert.

Table 8 LDAP Schema

Schema	Description
1. nsAddressTable	Address object. Stores addresses of people.
2. nsBillerProfile	Stores biller information.
3. nsCompanyProfile	Stores company information.
4. nsCustomerProfile	Stores membership information for BSP, Biller, and company.

Table 8 LDAP Schema

Schema	Description
5. nsDynamicGroup	Stores informationabout ojects that use dynamic groups. These are used for storing roles, privileges, and payment rules.
6. nsFinancialAccount	Stores information about the financial account.Used for both Biller receiving accounts and company's payment accounts.
7. nsPaymentType	Stores customer payment information.
8. nsPaymentVender	Stores information about payment vendors.

Enumerated Domains

Appendix A discusses the enumerated domains checking and the `enumdef.conf` configuration file.

Part 1

BillerXpert 4.6 Classes

Chapter 1

Base and Component Business Object Interfaces

This chapter discusses interfaces to base and core classes. These classes are the base classes for component business object interfaces or classes that provide a basic function, such as activity tracking or error handling, which is shared by all components in BillerXpert.

This chapter includes the following sections:

- Overview
- Base and Core Class Reference Descriptions

Overview

BillerXpert provides a class from which database records and contexts are derived. BillerXpert also provides several classes you can use to specify search criteria, as well as classes for activity tracking, manipulating configuration files, checking status information, and for generating unique IDs for database records. Table 1-1 lists these classes.

BXConfig Class Reference

The BXConfig class represent data stored in the BillerXpert configuration file, `billxpert.conf` and data stored for internationalization. The file is in name, type, value (NTV) format. See the *iPlanet BillerXpert 4.6 Customization Guide* for more information about this file.

Class Definition

Table 1-1 describes the BXConfig class definition.

Table 1-1 BXConfig Class Definition

Package	`netscape.billxpert.core;` `netscape.billxpert.core;`
Syntax	`public class BXConfig {...}`

Methods

This section describes the BXConfig class methods. Table 1-2 lists the BXConfig class methods. You only need to use the `getProperty()` and `setProperty()` methods from this class, which retrieve and modify, respectively, the value of a property in the file.

Table 1-2 BXConfig Class Methods

Method	Description
`getCollator()`	Retrieves the Collator for comparison for internationalization.
`getConFile()`	Retrieves the configuration file.
`getCountry()`	Retrieves the country code.
`getLanguage()`	Retrieves the given language for internationalization.
`getLocale()`	Retrieves the Locale for internationalization.
`getProperty()`	Retrieves the value of a property from the BillerXpert configuration file.
`init()`	Loads the configuration file.
`setCountry`	Modifies the country code.
`setLanguage`	Modifies the given language for internationalization.
`setProperty`	Modifies the value of a property in the BillerXpert configuration file.

getCollator()

Retrieves the collator.

Syntax	`public static Collator getCollator ();`

Parameters	None.
Discussion	Returns the Collator for locale independent comparisons.

getConfigFile()

Retrieves the configuration file.

Syntax	`public static String getConfigFile ();`
Parameters	None.
Discussion	Returns a string representing the BillerXpert configuration file name.

getCountry()

Retrieves the country code.

Syntax	`public static String getCountry();`
Parameters	None.
Discussion	The country code US and language would represent United States English. Returns a string representing the country code.

getLanguage()

Retrieves the language for internationalization.

Syntax	`public static String getLanguage();`
Parameters	None.
Discussion	The country code US and language code “en” would represent United States English. Returns a string representing the language.

getLocale()

Retrieves the locale.

Syntax	`public static String getLocale ();`

Parameters	None.
Discussion	Returns a string representing the locale for internationalization

getProperty()

Retrieves the value of a property from the BillerXpert configuration file.

Syntax	`public static String getProperty( String` *name* `) throws Exception;`
Parameters	*name* — A string that specifies the name of the property
Discussion	The `getProperty()` method loads the BillerXpert configuration file into memory if it is not already loaded and retrieves the value specified by the string from the file.

Example	The following example retrieves the database and user names from the configuration file:
	`try{`
	`String database_ =`
	`BXConfig.getProperty("database_section.database");`
	`String username_ =`
	`BXConfig.getProperty("database_section.username");`

setCountry()

Modifies the country for internationalization.

Syntax	`public static void setCountry ( String` *theCountry* `) throws Exception;`
Parameters	*theCountry* — A string that specifies the country.
Discussion	The country code US and language code “en” would represent United States English.

setLanguage()

Modifies the language for internationalization.

Syntax	`public static void setLanguage ( String` *theLanguage* `) throws Exception;`
Parameters	*theLanguage* — A string that specifies the language.
Discussion	The country code US and language code “en” would represent United States English.

setProperty()

Modifies the value of a property in the BillerXpert configuration file.

Syntax

```
public static void setProperty ( String name, String value ) throws Exception;
```

Parameters

name — A string that specifies the name of the property.

value — A string that specifies the new value.

Discussion

Loads the BillerXpert configuration file into memory if it is not already loaded and sets the specified property with the specified value. It writes the value to the configuration file and closes the file.

BXFilter Class Reference

The BXFilter class is used to represent a search filter, such as the Where clause of a SQL statement or a filter list for LDAP.

The BXFilter class supports the following items in a Where clause:

- AND
- Operators: >, >=, <, <=, =, !=, LIKE

Class Definition

Table 1-3 describes the BXFilter class definition.

Table 1-3 BXFilter Class Definition

Package	`netscape.billxpert.ejb.common;`
Syntax	`public class BXFilter {...}`

Methods

This section describes the methods of the BXFilter class. Table 1-4 lists the BXFilter class methods.

Table 1-4 BXFilter Class Methods *(1 of 2)*

Methods	Description
`addCondition()`	Adds a condition to a filter.
`generateWhereClause()`	Generates a where clause.
`getBXDBContext()`	Returns the billing context.
`getConditionSegment()`	Retrieves the condition segment.
`getParams()`	Returns the parameters generated by the `getWhereClause()`.
`getWhereClause()`	Sets up a where clause. Replaced by BXJDBCHelper class.
`resetWhereClause()`	Resets the Where clause.
`setBXDBContext`	Sets the billing context.
`toLdapFilter()`	Creates an LDAP filter list.

Table 1-4 BXFilter Class Methods *(2 of 2)*

Methods	Description
`toWhere()`	Creates an SQL Where clause.

Relational Operator Constants

The following constants are used to specify a relation between an attribute of a search and a value or expression:

Constants	Description

Syntax	`public static final String OP_GREATER = ">";`
	`public static final String OP_GREATEREQUAL = ">=";`
	`public static final String OP_LESS = "<";`
	`public static final String OP_LESSEQUAL = "<=";`
	`public static final String OP_EQUAL = "=";`
	`public static final String OP_NOTEQUAL = "!=";`
	`public static final String OP_LIKE = "LIKE";`
	`public static final String OP_IN = "IN";`
	`public static final String COND_AND = "AND";`
	`public static final String COND_OR = "OR";`

OP_GREATER	Greater than
OP_GREATEREQUAL	Greater than or equal to
OP_LESS	Less than
OP_LESSEQUAL	Less than or equal to
OP_EQUAL	Equal to
OP_NOTEQUAL	Not equal to
OP_LIKE	Like
COND_AND	Conditional AND
COND_OR	Conditional OR
OP_IN	Conditional IN.

addCondition()

Adds a condition to a filter.

Syntax

```
void addCondition( String attrName, String op, String value );
```

```
public void addCondition( String attrName, String op, String
value, String condition );
```

```
public void addCondition( BXFilter value, String condition );
```

Parameters

attrName — A string that specifies an attribute name.

op — A string that specifies a relational operation.

value — A string that specifes a value or an expression.

condition — A string that specifes a condition.

Discussion

The `addCondition()` method adds a condition to this filter. The condition evaluates to true or false and consists of the specified attribute name compared to the specified value or expression using the specified operator.

The attribute is case-insensitive. If the `value` parameter contains an expression, the expression is not evaluated by the `addCondition()` method.

Examples

The following example sets up a filter to select only records matching the object's primary key:

```
try {
I BXDocument doc = BXBillingMgr.createDocument("");
 doc.setStatus(status);
 doc.setBillingContext(billingContext);

 BXFilter filter = new BXFilter();
 Date today = new Date();
 Date from = new Date(today.getYear(),
 today.getMonth() - numMonth, today.getDate());

 String fromDate = BXDatetime.dateToDateString(from);
 String toDate = BXDatetime.dateToDateString(today);

 filter.addCondition("billdate", BXFilter.OP_GREATEREQUAL,
 fromDate);
 filter.addCondition("billdate", BXFilter.OP_LESSEQUAL,
 toDate);
 filter.addCondition("accountid", BXFilter.OP_EQUAL, acctid);

 docs = doc.ejbFindAll(filter, null);

}
```

generateWhereClause()

Generates a where clause.

Syntax	`public void generateWhereClause ( String` *`className`* `) throws Exception;`
Parameters	*className* — A string that specifies the class name used to retrieve the column and type in LDAP based on the class name.
Discussion	

getBXDBContext()

Returns the billing context.

Syntax	`public BXDBContext getBXDBContext();`
Parameters	None.
Discussion	Retrieves the billing context.

getConditionSegment()

Retrieves the condition segment.

Syntax	`public StringBuffer getConditionSegment (String` *`className`*`) throws Exception;`
Parameters	*className* — A string that specifies the class name used to in a get condition segment.
Discussion	Creates an SQL condition statement such as OP_IN or OP_LIKE SQL statements.

getParams()

Retrieves the parameters set by the `generateWhereClause()` method.

Syntax	`public IValList getParams();`
Parameters	None.
Discussion	Returns `params`

getWhereClause()

Sets up the Where clause. Replaced by the BXJDBCHelper class.

Syntax

```
public String getWhereClause();
```

Parameters

None.

Discussion

Returns `where.toString()`. The Where clause is generated by the `generateWhereClause()` method.

generateWhereClause()

Generates a where clause.

Syntax

```
public void generateWhereClause (String className) throws Exception;
```

Parameters

className — A string that specifies the class name used to retrieve the column and type in LDAP based on the class name.

Discussion

Generates the where clause.

resetWhereClause()

Sets up a where clause.

Syntax

```
public void resetWhereClause();
```

Parameters

None.

Discussion

Resets the where clause.

setBXDBContext()

Sets up the billing context.

Syntax

```
public void setBXDBContext (BXDBContext bx);
```

Parameters

bx — A BXDBContext hat represents the billing context.

Discussion

toLDAPFilter()

Creates an LDAP filter list.

Syntax

```
ACLdapFilterList toLDAPFilter( String className ) throws Exception;
```

Parameters

className — A string that specifies the class to use in the filter

Discussion

The value of the *className* parameter is used to retrieve an attribute. An exception is thrown if a filter attribute is not found for the specified class. Returns an ACLdapFilterList object.

Example

The following example retrieves all descendants of the specified class.

```
if ( filter != null ) {
LDAPfilter = filter.toLDAPFilter( className_ );
...
// get all descendants.
cltn = parent.getMyDescendants( dObj, LDAPfilter, LDAPorder,
LDAPasc, maxRows )
```

toWhere()

Creates an SQL Where clause.

Syntax

```
String toWhere( String className ) throws Exception;
```

Parameters

className - A string that specifies the class to use in the filter

Discussion

The Where clause created by this method does not contain the Where keyword. The value of the `className` parameter is used to retrieve an attribute. An exception is thrown if a filter attribute is not found for the specified class.

If more than one condition is specified in this filter, the `toWhere()` method concatenates each condition into the Where clause using logical AND operators. Returns a string that contains an SQL Where clause.

Example

```
public void ejbRemoveAll(BXFilter filter) throws Exception {
String where = null;
if(filter != null) {
try{
filter.setBXDBContext(context_);
//filter.generateWhereClause(getName());
where = filter.toWhere(getName());
} catch (Exception e) {
getBXStatus().push(BXStatus.STATUS_ERROR,BXError.BAD_FILTER,
null);
throw e;
}
}
stopWatch_.reset();
stopWatch_.start();
//Check if DBContext is valid
checkDBContext ();
context_.log (BXDBContext.INFO,
"BXEventHandler::ejbRemoveAll");
Connection conn = context_.getConnection ();
BXJDBCHelper helper = new BXJDBCHelper (conn);
helper.setBXDBContext (context_);
helper.setTable (getName () );
context_.log (BXDBContext.INFO,
"BXEventHandler::ejbRemoveAll:after getconnection");
//Execute the query
context_.log (BXDBContext.INFO,
"BXEventHandler::ejbRemoveAll;before deleteRow");
StringBuffer strbuf = new StringBuffer ("DELETE FROM " +
BXDBDef.BXEVENTHANDLER);
if (where != null )
```

```
public strbuf.append ("WHERE " + where);
Hashtable rh = null;
int rc = 0;
try {
helper.setTabledefTableName (getName () );
rh = helper.doExecute (strbuf.toString (),
BXJDBCHelper.DELETE);
rc = ((Integer) rh.get(new Integer
(BXDJDBCHelper.DELETE))).intValue();
} catch (SQLException se) {
}
}
```

BXOrder Class Reference

The BXOrder class represents order in which search results are returned, such as the Order By clause of an SQL statement or the order associated with attributes in an LDAP filter list.

Class Definition

Table 1-5 describes the BXOrder class definition.

Table 1-5 BXOrder Class Definition

Package	`netscape.billxpert.ejb.common;`
Syntax	`public class BXOrder {...}`

Order Specification Constants

The following constants are used to specify whether a record will be retrieved in ascending or descending order of the associated attribute:

Syntax	`public static final String ASC = "ASC";`
	`public static final String DSC = "DSC"`
	`ASC - Ascending order`
	`DSC - Descending order`

Methods

This section describes the BXOrder class methods. Table 1-6 lists these methods.

Table 1-6 BXOrder Class Methods *(1 of 2)*

Methods	Description
`addOrder()`	Associates an order with an attribute.
`resetOrderBy()`	Resets the SQL Order By clause.
`getSize()`	Determines the number of attributes for which an ordering has been specified.
`toLdapOrderBy()`	Creates an ordering for attributes in an LDAP filter list.

Table 1-6 BXOrder Class Methods *(2 of 2)*

Methods	Description
`toOrderBy()`	Creates an SQL Order By clause.
`setDBContext()`	Sets the database (DB) context.
`getDBContext()`	Retrieves the database (DB) context.

addOrder()

Associates an order with an attribute.

Syntax

```
public void addOrder (String attrName, String order);
```

Parameters

attrName — A string that specifies an attribute name.

order — A string that specifies the order.

Discussion

The `addOrder()` method adds a condition to this filter. The condition evaluates to true or false and consists of the specified attribute name compared to the specified value or expression using the specified operator. The attribute must exist in the `TypeDefs.conf` configuration file. If the `value` parameter contains an expression, the expression is not evaluated by the `addCondition()` method. For example:

```
order = new BXOrder();
order.addOrder( "CreateDate", BXOrder.DSC );
```

getSize()

Determines the number of attributes for which an ordering has been specified.

Syntax

```
public int getSize();
```

Parameters

None.

Discussion

Returns an integer that contains the name of the entity's class.

toLDAPOrderBy()

Creates an ordering for attributes in an LDAP filter list.

Syntax

```
void toLDAPOrderBy( String className, String[] order, boolean[] asc)
throws Exception;
```

Parameters

className — A string that specifies the class to which the ordering attributes belong.

order — An array of strings that spevifies the attributes on which to order the selected records.

asc — A string of booleans, each of which is true if the corresponding attribute is to be ordered in ascending order or is false if it is to be sorted in descending order

Discussion

Populates the *order* and boolean arrays with the values specified by calling the `addorder()` method. The elements of these arrays appear in the same order they were specified by calling the `addorder()` method. The value of the className parameter is used to retrieve an attribute. An exception is thrown if a filter attribute is not found for the specified class.

toOrderBy()

Creates an SQL Order By clause.

Syntax

```
String toOrderBy( String className ) throws Exception;
```

Parameters

className — A string that specifies the class to use in the filter.

Discussion

The attributes are returned as pairs consisting of an attribute and an ordering (ascending or descending); each pair is separated by a comma. The pairs appear in the same order as specified by calling the `addOrder()` method. The Order By clause created does not contain the Order By keywords.

The value of the *className* parameter is used to retrieve an attribute. An exception is thrown if an order by attribute is not found for the specified class. Returns a string that contains an SQL order by clause.

Example

```
String orderBy = null;
if( order != null ) {
try {
orderBy = order.toOrderBy( getName() );
} catch ( Exception e ) {
getBXStatus().push( BXStatus.STATUS_ERROR,
BXError.BAD_SORT, null);
throw e;
}
}
...
IQuery query = context_.createQuery();
...
if( orderBy != null )
query.setOrderBy( orderBy );
```

resetOrderBy()

Resets the Order By clause.

Syntax `public void resetOrderBy();`

Parameters None.

Discussion Reset the Order By clause.

setDBContext()

Sets the database context.

Syntax `public void set DBContext (BXDBContext bx);`

Parameters *bx* — A BXDBContext that specifies the database context.

Discussion

getDBContext()

Retrieves the database context.

Syntax `public void get DBContext (BXDBContext bx);`

Parameters	*bx* — A BXDBContext that specifies the database context.
Discussion	The database context.

BXStatus Class Reference

The BXStatus class represents a stack of status messages. You call the `push()` method to push an ID onto the object's stack. BillerXpert provides a BXStatus object you can use or you can create your own object.

If you use BillerXpert's BXStatus object, BillerXpert pops messages off the stack and processes them for you by replacing the ID with an actual message. You only need to use the following methods to retrieve information from the message stack.

Class Definition

Table 1-7 describes the BXStatus class definition.

Table 1-7 BXStatus Class Definition

Package	`netscape.billxpert.core;`
Syntax	`public class BXStatus {...}`

Constants

This section describes the constants contained in the BXStatus class. You may specify the kind of status to report by using a status constants.

Result Code Constants

Table 1-8 lists the result code constants.

Table 1-8 BXStatus Class Result Code Constants

Constant	Description

Syntax	`public static final int STATUS_OK = ACStatus.ACSTATUS_OK ;`
	`public static final int STATUS_INFO = ACStatus.ACSTATUS_INFO ;`
	`public static final int STATUS_WARNING = ACStatus.ACSTATUS_WARNING ;`
	`public static final int STATUS_ERROR = ACStatus.ACSTATUS_ERROR;`
	`public static final int STATUS_FATAL_ERROR = ACStatus.ACSTATUS_FATAL_ERROR;`

`STATUS_OK`	No error
`STATUS_INFO`	Informational message
`STATUS_WARNING`	Warning message
`STATUS_ERROR`	Error message
`STATUS_FATAL_ERROR`	Unknown fatal error

Name Constants

You may retrieve a value from the IValList associated with the status object by specifying the corresponding name constant.

Table 1-9 BXStatus Class Name Constants *(1 of 2)*

Constant	Description

Syntax	`public static final String RESULT_CODE = "resultCode";`
	`public static final String MESSAGE_CODE = "messageCode";`
	`public static final String MESSAGE_TYPE = "messageType";`
	`public static final String MESSAGE_FORMAT = "messageFormat";`
	`public static final String MESSAGE_TEXT = "messageText";`
	`public static final String EXCEPTION_DATA = "exceptionData";`
	`public static final String SOURCE_FILE = "sourceFile";`
	`public static final String SOURCE_LINE = "sourceLine";`

`RESULT_CODE`	Result code
`MESSAGE_CODE`	Message code
`MESSAGE_TYPE`	Message type

Table 1-9 BXStatus Class Name Constants *(2 of 2)*

Constant	Description
MESSAGE_FORMAT	Message format
MESSAGE_TEXT	Message text
RESULT_CODE	Result code
EXCEPTION_DATA	Exception data
SOURCE_FILE	Source file

Miscellaneous Constants

The following constants specify values in the status list that indicate whether the status information should be logged or displayed. Table 1-11 lists the miscellaneous constants.

Table 1-10 BXStatus Class Miscellaneous Constants

Constant	Description
Syntax	`public static final String IS_LOG = "isLog";` `public static final String IS_DISPLAY= "isDisplay";`
Context	`IS_LOG` —A string or object that is logged. `IS_DISPLAY` — A string or object that is displayed

Constructor

The BXStatus class contains the following constructor.

Syntax	`public BXStatus();` `public BXStatus (IValList[] list);`
Parameters	*list* — An array of IValList objects whose elements are error messages.
Discussion	Creates a BXStatus object. Use the first form of the constructor if you only need to specify the message when the object is pushed onto the stack. Use the second form of the constructor when you want to create an object that contains one or more messages.

Methods

This section describes the BXStatus class methods. Table 1-11 lists these methods.

Table 1-11 BXStatus Class Methods

Method	Description
`BXStatus()`	Creates a BXStatus object
`getBillerID()`	Specifies the BillerID
`setBillerID()`	Retrieves the BillerID
`push()`	Pushes a message on to the status object's message stack.
`pop()`	Pops the next message from the status object's message stack.
`top()`	Retrieve the first message from the status object's message stack.
`peek()`	Retrieves a message from the status object's message stack.
`process()`	Decodes and formats a message.
`count()`	Determines the number of messages in the status object's message stack.
`getACStatus ()`	Retrieves the ACStatus object
`dump()`	Pretty prints the status information
`clear()`	Remove all messages from in the status object's message stack.

clear()

Removes all messages from in the status object's message stack.

Syntax `public void clear();`

Parameters None.

Discussion

count()

Determines the number of messages in the status object's message stack.

Syntax	`public int count();`
Parameters	None.
Discussion	Returns an integer that returns the number of messages on the stack. If there are no messages on the stack, the count () method returns 0.

dump()

Pretty prints the status information.

Syntax	`public void dump ();`
Parameters	None.
Discussion	

getACStatus()

Retrieves the ACStatus object.

Syntax	`public ACStatus getACStatus ();`
Parameters	None.
Discussion	

getBillerID()

Retrieves the Biller ID.

Syntax	`public String BillerID ();`
Parameters	None.
Discussion	

peek()

Retrieves a message from the status object's message stack.

Syntax	`public IValList peek( int pos );`
Parameters	*pos* — An integer that specifies the position of message to retrieve.
Discussion	Returns an IValList object that contains the next message or null. The message stack is not modified; the pointer to the next message does not change. If a message does not exist at the specified position, null is returned. Messages begin at position 0.

pop()

Pops the next message from the status object's message stack.

Syntax

```
public void pop();
```

Parameters

None.

Discussion

The `pop()` method causes the top of the stack to change.

Example

The following method processes and pops all messages for a status object:

```
protected void logErrorMsg( BXStatus status ) {
if( status == null ) {
logOutput("Status is not set " );
return;
}
status.process();
IValList elm;
while ( (elm=status.top() )!= null) {
int rcd=elm.getValInt(BXStatus.RESULT_CODE);
String
msgcd=elm.getValString(BXStatus.MESSAGE_CODE);
String
msg=elm.getValString(BXStatus.MESSAGE_TEXT);
logOutput("Result code = "+rcd+",
Error number = "+msgcd+", error msg= "+msg );
status.pop(); // get rid of the processed one
}
)
```

```
protected void logErrorMsg( BXStatus status ) {
if( status == null ) {
logOutput("Status is not set " );
return;
}
status.process();
IValList elm;
while ( (elm=status.top() )!= null) {
int rcd=elm.getValInt(BXStatus.RESULT_CODE);
String
msgcd=elm.getValString(BXStatus.MESSAGE_CODE);
String
msg=elm.getValString(BXStatus.MESSAGE_TEXT);
logOutput("Result code = "+rcd+",
Error number = "+msgcd+", error msg= "+msg );
status.pop(); // get rid of the processed one
}
)
```

process()

Decodes and formats a message.

Syntax	`public void process();`
Parameters	None.
Discussion	See the example for the pop() method.

push()

Pushes a message on to the status object's message stack.

Syntax

```
public void push (int resultcd, String msgcd);
public void push (int resultcd, String msgcd, IValList parms);
public void push (int resultcd, String msgcd, IValList parms,
boolean display, boolean log);
public void push (int resultcd, int errCd, String errMsg,
String traceMsg, boolean display, boolean log);
```

Parameters

resultcd — A string that specifies the result code constant.

msgcd — A string that specifies the message code.

parms — An IValList object that specifies the message text.

traceMsg — A string that specifies an additional trace message.

display — A boolean value that specifies whether to display the message.

log — A boolean value that specifies whether to log the message

errCd — A string that specifies the error code.

errMsg — A string that specifies the error message.

Discussion

Use the first form of the `push()` method to both log and display the message. Use the second form the `push()` method to specify values for token replacement. Use the third form of the method to explicitly specify whether to log the message or display the message.

Example

The following example pushes a message onto the BillerXpert defined status object's stack:

```
if (request == null) {
status.push(BXStatus.STATUS_ERROR, BXErrorCode.
AL_INVALID_REQUEST);
}
```

The following example pushes a message with one token onto the BillerXpert defined status object's stack:

```
IValList paramList = GX.CreateValList();
paramList.setValString("function", "ejbFindAll" );
getBXStatus().push( BXStatus.STATUS_ERROR,
BXError.NOT_IMPLEMENTED,
paramList );
```

The message associated with `BXError.NOT_IMPLEMENTED` is:

```
"EJB_000018" Str "[@function@] is not implemented."
```

Chapter 2

Billing Classes Component Business Object Interfaces

This chapter discusses component business object interface classes for billing records. These classes provide access to records stored in an Oracle database.

This chapter includes the following sections:

- Overview
- Billing Class Reference Descriptions

Overview

Billing classes represent billing records in the database. You use these classes to manipulate these records. Table 2-1 lists the classes.

Table 2-1 Billing Classes *(1 of 3)*

Class Name	Description	Location
1. BSPBillerInfo		page 59
2. IBSPBillerInfo		page 59
3. BXBillerCustAcct	Base class that manipulates BillerXpert customer account objects.	page 59
4. IBXBillerCustAcct	Interface that manipulates BillerXpert customer account objects.	page 59
5. BXBillingContext	Communicate between an AppLogic and an EJB component.	page 81
6. BXBillingEntity	Base class for billing business objects.	page 83

Table 2-1 Billing Classes *(2 of 3)*

Class Name	Description	Location
7. IBXBillingEntity	Interface for billing business objects.	page 83
8. BXBillingMgr	Creates billing-related component business objects.	page 87
9. BXCardInfo	Base class for credit card information.	page 91
10. IBXCardInfo	Interface for credit card information.	page 91
11. BXClickThrough	Base class that manipulates click through objects.	page 103
12. IBXClickThrough	Interface that manipulates click through objects.	page 103
13. BXDetailCall	Base class that manipulates detail call objects	page 103
14. IBXDetailCall	Interface that manipulates detail call objects	page 103
15. BXDetailMisc	Base class that manipulates detail miscellaneous objects	page 103
16. IBXDetailMisc	Interface that manipulates detail miscellaneous objects	page 103
17. BXDetailRating	Base class that manipulates detail rating objects	page 103
18. IBXDetailRating	Interface that manipulates detail rating objects	page 103
19. BXDocument	Base class that manipulates billing objects.	page 103
20. IBXDocument	Interface that manipulates billing objects.	page 104
21. BXOFXSignOn	Base class that manipulates OFX sign on objects.	page 121
22. IBXOFXSignOn	Interface that manipulates OFX sign on objects.	page 121
23. BXPayHistory	Manipulates payment history objects.	page 122
24. IBXPayHistory	Manipulates payment history objects.	page 122
25. BXPaySchedule	Base class that manipulates payment schedule objects.	page 122
26. IBXPaySchedule	Interface that manipulates payment schedule objects.	page 122
27. BXSummary	Base class that manipulates billing summary objects.	page 123
28. IBXSummary	Interface that manipulates map objects.	page 123
29. CSBXMap	Base class that manipulates map objects.	page 144
30. ICSBXMap	Interface that manipulates billing summary objects.	page 146

Table 2-1 Billing Classes *(3 of 3)*

Class Name	Description	Location
31. OFXRequestLog	Base class that manipulates OFX request log objects.	page 146
32. IOFXRequestLog	Interface that manipulates OFX request log objects.	page 146

BSPBillerInfo Class Reference

IBSPBillerInfo Class Reference

BXBillerCustAcct Class Reference

The BXBillerCustAcct class is the base class for defining methods that manipulate objects representing customer account records. It also defines methods that allow access to the account and related records.

Whenever possible, use the interface class IBXBillerCustAcct (see “IBXBillerCustAcct Class Reference) instead of BXBillerCustAcct.

IBXBillerCustAcct Class Reference

The IBXBillerCustAcct interface class manipulates BillerXpert customer account objects.

Class Definition

Table 2-2 describes the IBXBillerCustAcct interface definition.

Table 2-2 IBXBillerCustAcct Interface Definition

Package	`netscape.billxpert.ejb.billing;`
Syntax	`public interface IBXBillerCustAcct extends IBXBillingEntity {...}`

Methods

This section describes the IBXBillerCustAcct class methods. Table 2-3 lists these methods.

Before you can call the `IBXEntity.ejbCreate()` (see "IBXBillingEntity Class Reference," on page 83) method to create a new record associated with the object, you must call the `setBillerID()`, `setUserID()`, and `setAccountNumber()` methods to set the biller ID, user ID, and account number fields, respectively.

Table 2-3 IBXBillerCustAcct Methods *(1 of 4)*

Methods	Description
Manipulating customer account and related records	
ejbAssociateUser()	Retrieves a customer account record by account name.
`ejbFindAllDocuments()`	Retrieves documents associated with a customer account.
`ejbFindAllContainers()`	Retrieves containers associated with a customer account.
`ejbFindAllBCAs()`	Retrieves all customer accounts registered to a user.
`ejbFindAllUserIDs()`	Retrieves all user IDs associated with a customer account.
`ejbFindPaymentSchedule()`	Retrieves the payment schedule for a customer account.
`ejbFindByAccountNumber()`	Associates an account with a user.
`ejbRemoveAssociatedUser()`	Disassociates an account from a user.
`ejbFindAllPaymentHistories()`	Retrieves payment history for a customer account
`ejbFindAllAutoPaySchedules`	Retrieves autopay schedules for a customer account
Getting and setting customer account information	
`getBillerID()`	Retrieves the biller ID.
`setBillerID()`	Specifies the biller ID.
`getAccountType()`	Retrieves the customer's account type.
`setAccountType()`	Specifies the customer's account type.
`getCustomerType()`	Retrieves the customer type.
`setCustomerType()`	Specifies the customer type.
`getDescription()`	Retrieves the customer's description.
`setDescription()`	Specifies the customer's description.
`getTaxID()`	Retrieves the customer's tax ID.

Table 2-3 IBXBillerCustAcct Methods *(2 of 4)*

Methods	Description
`setTaxID()`	Specifies the customer's tax ID.
`getSince()`	Retrieves the year in which the account was established.
`setSince()`	Specifies the year in which the account was established.
`getServiceOptions()`	Retrieves the account's service options.
`setServiceOptions()`	Specifies the account's service options.
`getProfile()`	Retrieves the profile.
`setProfile()`	Specifies the profile.
`getAccountNumber()`	Retrieves the customer's account number.
`setAccountNumber()`	Specifies the customer's account number.
`getAccountStatus()`	Retrieves the account status.
`setAccountStatus()`	Specifies the account status.
`getUnpaidBills()`	Retrieves the quantity of unpaid bills.
`setUnpaidBills()`	Specifies the quantity of unpaid bills.
`getUnpaidBalance()`	Retrieves the unpaid balance.
`setUnpaidBalance()`	Specifies the unpaid balance.
`getAccountBalance()`	Retrieves the account balance.
`setAccountBalance()`	Specifies the account balance.
`getAccountBalanceDate()`	Retrieves the balance due date.
`setAccountBalanceDate()`	Specifies the balance due date.
`getCreateDate()`	Retrieves the customer account's creation date.
`getModifyDate()`	Retrieves the customer account's modification date.
`getDisputeStatus()`	Retrieves the dispute status.
`setDisputeStatus()`	Specifies the dispute status.
`getExtension()`	Retrieves the customer's extension.
`setExtension()`	Specifies the customer's extension.
`getUserID()`	Retrieves the user ID.
`setUserID()`	Specifies the user ID.
`getBillingAddress()`	Retrieves the customer's billing address.
`setBillingAddress()`	Specifies the customer's billing address.

Table 2-3 IBXBillerCustAcct Methods *(3 of 4)*

Methods	Description
`getServiceAddress()`	Retrieves the customer's service address ID.
`setServiceAddress()`	Specifies the customer's service address ID.
`getVerifyUser()`	Retrieves the user.
`setVerifyUser()`	Specifies the user.
`getAuthorizedUser()`	Retrieves the authorized user.
`setAuthorizedUser()`	Specifies the authorized user.
`getAuthorizedDate()`	Retrieves the authorized date.
`setAuthorizedDate()`	Specifies the authorized date.
`getVerifiedDate()`	Retrieves the verification date.
`setVerifiedDate`	Specifies the verification date.
`getFileID()`	Retrieves the file ID.
`setFileID()`	Specifies the file ID.
`getAutopay()`	Retrieves the autopay flag.
`setAutopay()`	Specifies the autopay flag.
`getExtDate1`	Retrieves the first extension date.
`setExtDate1`	Specifies the first extension date.
`getExtDate2()`	Retrieves the second extension date.
`setExtDate2()`	Specifies the second extension date.
`getExtDate3()`	Retrieves the third extension date
`setExtDate3()`	Specifies the third extension date.
`getExtDate4()`	Retrieves the fourth extension date
`setExtDate4()`	Specifies the fourth extension date.
`getExtNum1()`	Retrieves the first extension number.
`setExtNum1()`	Specifies the first extension number.
`getExtNum2()`	Retrieves the second extension number.
`setExtNum2()`	Specifies the second extension number.
`getExtNum3()`	Retrieves the third extension number.
`setExtNum3()`	Specifies the third extension number.
`getExtNum4()`	Retrieves the fourth extension number.

Table 2-3 IBXBillerCustAcct Methods *(4 of 4)*

Methods	Description
`setExtNum4()`	Specifies the fourth extension number.
`getCompanyID()`	Determines Company ID.
`setCompanyID()`	Specifies Company ID.

ejbAssociateUser()

Associates an account with a user.

Syntax

```
public void ejbAssociateUser( String acctNo, String
userID, String companyID) throws Exception;
```

Parameters

acctNo — A string that specifies the account number.

userID — A string that specifies the user ID.

companyID — String that specifies the company ID.

Discussion

Retrieves the customer account record identified by the account number from the database and specifies the fields of this object from the contents of the record. It then associates the specified account number with the specified user and updates the association in the database.

The customer account's database context must previously have been set by calling this object's `createBillingContext()` method. The customer account record identified by the account number must already exist in the database. The `ejbAssociateUser()` method can be committed or rolled back as part of a transaction.

ejbFindAllBCAs()

Retrieves all customer accounts registered to a user.

Syntax

```
public Vector ejbFindAllBCAs( String companyID ) throws
Exception;
```

```
public Vector ejbFindAllBCAs( String userID, String
companyID ) throws Exception;
```

Parameters

userID — String that specifies the user ID.

companyID — String that specifies the company ID.

Discussion	Retrieves all customer account records for the customer specified by user ID. The customer account object's database context must previously have been set by calling each object's `createBillingContext()` method. For each record, the `ejbFindAllBCA()` method creates an object, specifies the fields in the object from a retrieved record, and adds the object to the return vector. Returns a vector that contains `BXBillerCustAcct` objects representing the selected records.

ejbFindAllContainers()

Retrieves containers associated with a customer account.

Syntax	`public Vector ejbFindAllContainers( BXFilter` *filter*`, BXOrder` *order* `) throws Exception;`
Parameters	*filter* — BXFilter object that specifies the Where clause of an SQL Select statement. *order* — BXOrder that specifies the Order By clause of an SQL Select statement
Discussion	The `ejbFindAllContainers()` method retrieves all container records for the account that match the clause in the *filter* parameter, ordered by the clause specified in the order parameter. For each record, the `ejbFindAllContainers()` method creates an object, specifies the fields in the object from the record, and adds the object to the return vector. If the *filter* parameter is null, all records will be retrieved. If the `order` parameter is `null`, the retrieved records will be sorted in ascending order by their primary key. The customer account object's database context must previously have been set by calling each object's `createBillingContext()` method. The account ID must be set in the customer account object.

ejbFindAllDocuments()

Retrieves all documents associated with a customer account.

Syntax	`public Vector ejbFindAllDocuments( BXFilter` *filter*`, BXOrder` *order*`) throws Exception;`

Parameters	*filter* — BXFilter object that specifies the Where clause of an SQL Select statement. If filter is null, all records will be retrieved. *order* — BXOrder that specifies the Order By clause of an SQL Select statement. If order is null, the retrieved records are sorted in ascending order by their primary key.
Discussion	Retrieves all document records for an account that match the clause in the *filter* parameter, ordered by the clause specified in the order parameter. For each record, `ejbFindAllDocuments()` creates an object, specifies the fields in the object from the record, and adds the object to the return vector. The customer account object's database context must previously have been set by calling each object's `createBillingContext()` method. The account ID must be set in the customer account object. Returns a vector that contains `BXDocument` objects representing the selected records.

ejbFindAllUserIDs()

Retrieves all user IDs associated with a customer account.

Syntax	`public Vector ejbFindAllUserIDs( String` *acctNo*`) throws Exception;`
Parameters	*acctNo* — String that specifies the account number
Discussion	Returns a vector that contains strings, each of which is a user ID. The customer account's database context must previously have been set by calling this object's `createBillingContext()` method.

ejbFindByAccountNumber()

Retrieves a customer account record by account name.

Syntax	`public void ejbFindByAccountNumber( String` *acctNo*`, String` *biller* `) throws Exception;`
Parameters	*acctNo* — A string that specifies the account number. *biller* — A string that specifies the account name

Discussion	Retrieves the customer account record specified by the account number. The object's fields are set from the contents of the record. The customer account's database context must previously have been set by calling this object's `createBillingContext()` method. The values of fields already set in the object are overwritten by this method; values in the database are not affected.

ejbFindPaymentSchedule()

Retrieves the payment schedule for a customer account.

Syntax	`public IBXPaySchedule ejbFindPaymentSchedule() throws Exception`
Parameters	None.
Discussion	Returns thehe `IBXPaySchedule` object associated with this customer account or `null` if no `IBXPaySchedule` object is associated with the account. The customer account object's database context must previously have been set by calling each object's `createBillingContext()` method. The account ID must be set in the customer account object.

ejbRemoveAssociatedUser()

Disassociates an account from a user.

Syntax	`public void ejbRemoveAssociatedUser( String` *`acctNo`*`, String` *`userID`*`) throws Exception;`
Parameters	*acctNo* — A string that specifies the account number *userID* — A string that specifies the user ID
Discussion	Removes the association between the specified account number and the specified user from the database. The association must already exist in the database. The customer account's database context must previously have been set by calling this object's `createBillingContext()` method. The `ejbAssociateUser()` method can be committed or rolled back as part of a transaction

ejbFindAllPaymentHistories()

Retrieves payment history for a customer account.

Syntax

```
public Vector ejbFindAllPaymentHistories (BXfilter filter,
BXOrder order) throws Exception;
```

Parameters

filter — Parameter for selecting the appropriate history

order — Sequence in payment history

Discussion

Finds the association between the specified account number and the specified user from the database. The association must already exist in the database. The customer account's database context must previously have been set by calling this object's `createBillingContext()` method.

The `ejbAssociateUser()` (see page 63) method can be committed or rolled back as part of a transaction.

ejbFindAllAutoPaySchedules()

Retrieves autopay schedules for a customer account.

Syntax

```
public Vector ejbFindAllPaymentHistories (BXfilter filter,
BXOrder order) throws Exception;
```

Parameters

filter — A parameter for selecting the appropriate history.

order — The sequence in payment scheduling.

Discussion

Finds the association between the specified account number and the specified user from the database. The association must already exist in the database. The customer account's database context must previously have been set by calling this object's `createBillingContext()` method. The `ejbAssociateUser()` method can be committed or rolled back as part of a transaction

getAccountBalance()

Retrieves the account balance.

Syntax

```
public double getAccountBalance();
```

Parameters

None.

Discussion

Returns a double-precision floating number that contains the balance.

getAccountBalanceDate()

Retrieves the balance due date.

Syntax	`public Date getAccountBalanceDate();`
Parameters	None.
Discussion	Returns the `Date` object that contains the date

getAccountNumber()

Retrieves the customer's account number.

Syntax	`public String getAccountNumber();`
Parameters	None.
Discussion	Returns a string that contains the account number.

getAccountStatus()

Retrieves the account status.

Syntax	`public String getAccountStatus();`
Parameters	None.
Discussion	Returns a string that contains the account status.

getAccountType()

Retrieves the customer's account type.

Syntax	`public String getAccountType();`
Parameters	None.
Discussion	Returns a string that contains the account type.

getBillerID()

Retrieves the biller ID.

Syntax	`public String getBillerID();`

Parameters	None.
Discussion	Returns a string that contains the ID.

getBillingAddress()

Retrieves the customer's billing address.

Syntax	`public String getBillingAddressID();`
Parameters	None.
Discussion	Returns a string that contains the address.

getCustomerType()

Retrieves the customer's account type.

Syntax	`public String getCustomerType();`
Parameters	None.
Discussion	Returns a string that contains the customer type.

getDescription()

Retrieves the customer's description.

Syntax	`public String getDescription();`
Parameters	None.
Discussion	Returns a string that contains the customer description.

getDisputeStatus()

Retrieves the dispute status.

Contents	**Description**
Syntax	`public String getDisputeStatus();`

Contents	Description
Parameters	None.
Discussion	Returns a string that contains the status.

getExtension()

Retrieves the customer's extension.

Contents	Description
Syntax	`public String getExtension();`
Parameters	None.
Discussion	Returns a string that contains the extension.

getModifyDate()

Retrieves the customer account's modification date.

Syntax	`public Date getModifyDate();`
Parameters	None.
Discussion	Returns a Date object that contains the modification date.

getProfile()

Retrieves the profile.

Syntax	`public String getProfile();`
Parameters	None.
Discussion	Returns a string that contains the profile.

getServiceAddressID()

Retrieves the customer's service address ID.

Syntax	`public String getServiceAddressID();`
Parameters	None.

Discussion	Returns a string that contains the address.

getServiceOptions()

Retrieves the account's service options.

Syntax	`public String getServiceOptions();`
Parameters	None.
Discussion	Returns a string that contains the service options.

getSince()

Retrieves the year in which the account was established.

Syntax	`public int getSince();`
Parameters	None.
Discussion	Returns an integer that contains the year.

getTaxID()

Retrieves the customer's tax ID.

Syntax	`public String getTaxID()`
Parameters	None.
Discussion	Returns a Date object that contains the tax identification.

getUnpaidBalance()

Retrieves the unpaid balance.

Syntax	`public double getUnpaidBalance();`
Parameters	None.
Discussion	Returns a double-precision floating number that contains the balance.

getUnpaidBills()

Retrieves the quantity of unpaid bills.

Syntax	`public int getUnpaidBills();`
Parameters	None.
Discussion	Returns an integer that contains the quantity.

getUserID()

Retrieves the user ID.

Syntax	`public String getUserID();`
Parameters	None.
Discussion	Returns a string that contains the ID.

getVerifiedUser()

Retrieves the verification date.

Syntax	`public String getVerifiedUser();`
Parameters	None.
Discussion	Returns a string that verified user.

setAuthorizedUser()

Retrieves the authorization of an user.

Syntax	`public String getAuthorizedUser();`
Parameters	None.
Discussion	Returns a string that authorizes the user.

getAuthorizedDate()

Retrieves the authorization date.

Syntax	`public String getAuthorizeDate();`

Parameters	None.
Discussion	Returns a string that authorizes the date.

getVerifiedDate()

Retrieves the verification date.

Syntax	`public String getVerifiedDate();`
Parameters	None.
Discussion	Returns a string that verifies the authorization date.

getFileID()

Retrieves the file ID.

Syntax	`public String getFileID();`
Parameters	None.
Discussion	Returns a string that verified the fileID.

getAutoPay()

Retrieves the autopay amount.

Syntax	`public String getAutoPay();`
Parameters	None.
Discussion	Returns a string that verifies the autopay amount.

setAccountBalance()

Specifies the account balance.

Syntax	`public void setAccountBalance( double` *acctBalance*`);`
Parameters	*acctBalance* — A double-precision floating point number that specifies the balance
Discussion	Only changes the value of the account balance in the object; it does not modify the value in the database.

setAccountBalanceDate()

Specifies the account balance due date.

Syntax	`public void setAccountBalanceDate( Date` *acctBalanceDate*`);`
Parameters	*acctBalanceDate* — A Date object that specifies the date the account balance is due.
Discussion	Only changes the value of the account balance date in the object; it does not modify the value in the database.

setAccountNumber()

Specifies the customer's account number.

Syntax	`public void setAccountNumber( String` *acctNum*`);`
Parameters	*acctNumber* — A string that specifies the account number
Discussion	Only specifies of the account number in the object; it does not modify the value in the database.

setAccountStatus()

Specifies the account status.

Syntax	`public void setAccountStatus( String` *acctStatus*`);`
Parameters	*acctStatus* — A string that specifies the account status.
Discussion	Only specifies of the account status in the object; it does not modify the value in the database.

setAccountType()

Specifies the customer's account type.

Syntax	`public void setAccountType( String` *acctType*`) ;`
Parameters	*acctType* — Specifies the account type
Discussion	Only specifies of the account type in the object; it does not modify the value in the database.

setBillerID()

Specifies the biller ID.

Syntax	`public void setBillerID( String` *billerID*`);`
Parameters	*billerID* — A string that specifies the biller ID
Discussion	Only specifies of the biller ID in the object; it does not modify the value in the database.

setBillingAddress()

Specifies the customer's billing address.

Syntax	public void setBillingAddressID (String *addressID*);
Parameters	*addressID* — A string that specifies the biller address ID
Discussion	Only specifies of the biller address ID in the object; it does not modify the value in the database.

setCustomerType()

Specifies the customer type.

Syntax	`public void setCustomerType( String` *CustomerType* `);`
Parameters	*customerType* — String that specifies the customer type
Discussion	Only specifies of the customer type in the object; it does not modify the value in the database.

setDescription()

Specifies the customer's description.

Syntax	`public void setDescription( String` *description*`);`
Parameters	*description* — A string that specifies the description
Discussion	Only specifies of the description of the object; it does not modify the value in the database.

setDisputeStatus()

Specifies the dispute status.

Syntax	`public void setDisputeStatus( String` *disputeStatus*`);`
Parameters	*disputeStatus* — String that specifies the dispute status
Discussion	Only specifies of the dispute status of the object; it does not modify the value in the database.

setExtension()

Specifies the customer's extension.

Syntax	`public void setExtension ( String` *extension*`);`
Parameters	*extension* — String that specifies the extension
Discussion	Only specifies of the extension of the object; it does not modify the value in the database.

setProfile()

Specifies the profile.

Syntax	`public void setProfile( String` *profile*`);`
Parameters	*profile* — String that specifies the profile.
Discussion	Only specifies of the profile of the object; it does not modify the value in the database.

setServiceAddressID()

Specifies the customer's service address ID.

Syntax	`public void setServiceAddressID ( String` *addressID* `);`
Parameters	*addressID* — String that specifies the address ID
Discussion	Only specifies of the service ID in the object; it does not modify the value in the database.

setServiceOptions()

Specifies the account's service options.

Syntax	`public void setServiceOptions( String` *serviceOption*`);`
Parameters	*serviceOption* — A string that specifies the service option
Discussion	Only specifies of the service option in the object; it does not modify the value in the database.

setSince()

Specifies the year in which the account was established.

Syntax	`public void setSince ( int` *Since* `);`
Parameters	*since* — An integer that specifies the year.
Discussion	Only specifies of the value change field in the object; it does not modify the value in the database.

setTaxID()

Specifies the customer's tax ID.

Syntax	`public void setTaxID( String` *taxID*`);`
Parameters	*taxID* — A string that specifies the tax ID
Discussion	Only specifies of the tax ID in the object; it does not modify the value in the database.

setUnpaidBalance()

Specifies the unpaid balance.

Syntax	`public void setUnpaidBalance ( double` *unpaidBalance* `);`
Parameters	*unpaidBalance* — String that specifies the unpaid balance
Discussion	Only specifies of the unpaid balance in the object; it does not modify the value in the database.

setUnpaidBills()

Specifies the quantity of unpaid bills.

Syntax	`public void setUnpaidBills( int` *unpaidBills* `);`
Parameters	*unpaidBills* — String that specifies the unpaid balance
Discussion	Only specifies of the unpaid balance in the object; it does not modify the value in the database.

setUserID()

Specifies the user ID.

Syntax	`public void setUserID( String userID);`
Parameters	*userID* — A string that specifies the user ID
Discussion	Only specifies of the user ID in the object; it does not modify the value in the database.

getExtDate1()

Retrieves the first extension date.

Syntax	`public Date getExtDate1();`
Parameters	None.
Discussion	

setExtDate1()

Specifies the credit card first extension date.

Syntax	`public void setExtDate1 (Date` *date*`);`
Parameters	*date* — The first extension date.
Discussion	

getExtDate2()

Retrieves the second extension date.

Syntax `public Date getExtDate2();`

Parameters None.

Discussion

setExtDate2()

Specifies the second extension date.

Syntax `public void setExtDate1 (Date` *date*`);`

Parameters *date* — The second extension date.

Discussion

getExtDate3()

Retrieves the third extension date.

Syntax `public Date getExtDate3();`

Parameters None.

Discussion

setExtDate1()

Specifies the third extension date.

Syntax `public void setExtDate3 (Date` *date*`);`

Parameters *date* — The third extension date.

Discussion

getExtDate4()

Retrieves the fourth extension date.

Syntax `public Date getExtDate4();`

Parameters None.

Discussion

setExtDate4()

Specifies the fourth extension date.

Syntax `public void setExtDate4 (Date` *date*`);`

Parameters *date* — The fourth extension date.

Discussion

getExtNum1()

Retrieves the first extension number.

Syntax `public double getExtNum1();`

Parameters None.

Discussion

setExtNum1()

Specifies the first extension number.

Syntax `public void setExtNum1 (double` *num*`);`

Parameters *num* — The first extension number.

Discussion

getExtNum2()

Retrieves the second extension number.

Syntax `public double getExtNum2();`

Parameters None.

Discussion

setExtNum2()

Specifies the second extension number.

Syntax `public void setExtNum2 (double num);`

Parameters *num* — The second extension number.

Discussion

getExtNum3()

Retrieves the third extension number.

Syntax `public double getExtNum3();`

Parameters None.

Discussion

setExtNum3()

Specifies the third extension number.

Syntax `public void setExtNum3 (double num);`

Parameters *num* — The third extension number.

Discussion

getExtNum4()

Retrieves the fourth extension number.

Syntax `public double getExtNum4();`

Parameters None.

Discussion

getCompanyID()

Determines the Company ID.

Syntax	`public String getCompanyID();`
Parameters	None.
Discussion	Returns the Company ID associated with the current context

setCompanyID()

Specifies the Company ID.

Syntax	`public void setCompanyID (String` *companyID*`);`
Parameters	*CompanyID* — A string that specifies the company ID.
Discussion	Returns a reference to the status object.

BXBillingContext Class Reference

The BXBillingContext class is used to associate database related parameters or information with the billing EJBs. BXBillingContext extends the BXContext class, which is accessed via the IBXDBContext interface (see "IBXDBContext Class Reference," on page 72).

Instances of the BXBillingContext class represent a database context for billing objects. Typically you create a billing object using one of the BXBillingMgr class methods (see "BXBillingMgr Class Reference," on page 87) and then create a context for it by creating a BXBillingContext object.

Class Definition

Table 2-2 describes the BXBillingContext class definition.

Table 2-4 BXBillingContext Class Definition

Package	`netscape.billxpert.ejb.billing;`
Syntax	`public class BXBillingContext extends BXDBContext {...}`

Constructor

This section describes the BXBillingContext constructor. Table 2-5 lists this constructor.

Table 2-5 BXBillingContext Constructors

Constructor	Description
`BXBillingContext()`	Creates a BXBillingContext object. (Note: there are four instances of the BXBillingContext constructors.)

BXBillingContext()

This constructor creates a BXBillingContext object.

Syntax

```
public BXBillingContext (IContext context, String billerID)
throws Exception;

public BXBillingContext (IContext context, BXStatus status,
String billerId) throws Exception;

public BXBillingContext (IContext context, BXStatus status,
String billerId, String companyId) throws Exception;

public BXBillingContext (IContext context, String datasource,
String database, String username, String password, int trace);
```

Parameters

context — An `IContext` object that specifies the context.

status — A `BXStatus` object that holds the current status of the context.

datasource — A string that specifies the data source.

database — A string that specifies the database.

username — A string that specifies the user name.

password — A string that specifies the password.

billerId — A string that specifies the biller id.

companyId — A string that specifies the company id.

trace — An integer that specifies the debug trace level.

Discussion	You typically create a billing context object by specifying the applogic's *context* variable for the context parameter. The first and third forms of the constructor do not associate a status object with the context. In this case you, call the `setStatus()` or `getBXStatus()` method to specify a status object for the context. The first two forms of the constructor take the database login information from a configuration file. If you use the third form of the constructor, specify the same value for the `datasource` parameter as you do for the database parameter.

BXBillingEntity Class Reference

The BXBillingEntity class is the base class for billing business objects.

Whenever possible, use the interface class IBXBillingEntity (see "IBXBillingEntity Class Reference) instead of BXBillingEntity.

IBXBillingEntity Class Reference

The IBXBillingEntity class is the interface for the Billing Entity bean for all billing classes. You must call the `setBillingContext()` method of each object that represents a billing record in the database before you can manipulate the record with the objects.

Class Definition

Table 2-6 describes the IBXBillingEntity interface definition.

Table 2-6 BXBillingEntity Interface Definition

Package	`netscape.billxpert.ejb.billing;`
Syntax	`public interface IBXBillingEntity extends IBXEntity {...}`

Methods

This section describes the IBXBillingEntity interface methods. Table 2-7 lists these methods.

Table 2-7 IBXBillingEntity Interface Methods

Methods	Description
Getting and setting billing context information	
`getBillingContext()`	Retrieves the billing context.
`setBillingContext()`	Specifies the billing context.
`getExtension1()`	Retrieves the extension1 context.
`setExtension1()`	Specifies the extension1 context.
`getExtension2()`	Retrieves the extension2 context.
`setExtension2()`	Specifies the extension2 context.
`getExtension3()`	Retrieves the extension3 context.
`setExtension3()`	Specifies the extension3 context.
`getExtension4()`	Retrieves the extension4 context.
`setExtension4()`	Specifies the extension4 context.

getBillingContext()

Retrieves the billing context.

Syntax	`public BXBillingContext getBillingContext();`
Parameters	None.
Discussion	Returns the BXBillingContext object associated with this billing object.

getExtension1()

Retrieves the first extension context.

Contents	Description
Syntax	public String getExtension1();
Parameters	None.
Discussion	Returns the `BXExtension1` object.

getExtension2()

Retrieves the second extension context.

Syntax	`public String getExtension2();`
Parameters	None.
Discussion	Returns the `BXExtension2` object.

getExtension3()

Retrieves the third extension context.

Syntax	`public String getExtension3();`
Parameters	None.
Returns	The `BXExtension3` object.

getExtension4()

Retrieves the fourth extension context.

Syntax	`public String getExtension4();`
Parameters	None.
Discussion	Returns the `BXExtension4` object.

setBillingContext()

Specifies the billing context.

Syntax	`public void setBillingContext ( BXBillingContext` *context* `);`
Parameters	*context* — The context associated with this billing object.
Discussion	

setExtension1()

Specifies the first extension context.

Syntax	`public void setExtension1( String` *value* `);`

Parameters	*value* — The context associated with the first extension object.
Discussion	

setExtension2()

Specifies the second extension context.

Contents	**Description**
Syntax	`public void setExtension2( String value );`
Parameters	*value* — The context associated with the second extension object.
Discussion	

setExtension3()

Specifies the third extension context.

Syntax	`public void setExtension3 (String value);`
Parameters	*value* — The context associated with the third extension object.
Discussion	

setExtension4()

Specifies the fourth extension context.

Syntax	`public void setExtension4( String value );`
Parameters	*value* — The context associated with fourth extension object.
Discussion	

BXBillingMgr Class Reference

The BXBillingMgr class defines methods that create billing objects. You typically create a billing object and use it to manipulate a record in the database.

Class Definition

Table 2-8 describes the BXBillingMgr class definition.

Table 2-8 BXBillingMgr Class Definition

Package	`netscape.billxpert.ejb.billing;`
Syntax	`public class BXBillingMgr {...}`

Methods

This section describes the BXBillingMgr class methods. Table 2-8 lists these methods.

Table 2-9 BXBillingMgr Method Definitions

Methods	Description
Create billing objects	
`createBillerCustAcct()`	Creates a biller customer account object.
`createContainer()`	Creates a container object.
`createDocument()`	Creates a document object.
`createSummary()`	Creates a summary object.
`createPaySchedule()`	Creates a payment schedule object.
`createPayHistory()`	Creates a payment history object.
`createEventLog()`	Creates an event log object.
`createMessage()`	Creates a message object.
`createCardInfo()`	Creates a credit card object
Create entity object	
`createObject()`	Creates an object.

createBillerCustAcct()

Creates a biller customer account object.

Syntax	`public static IBXBillerCustAcct createBillerCustAcct( String` *key*`) throws Exception;`

Parameters	*key* — A string that contains the object mapping key in the format of `billerName.acctType`.
Discussion	The `createBillerCustAcct` method is a class method. If you use only the default account, specify null for the billName acctType. Returns an IBXBillerCustAcct object.

createCardInfo()

Creates a credit card object.

Syntax	`public static IBXCardInfo createCardInfo ( String` *key*`) throws Exception;`
Parameters	*key* — A string that contains the object mapping key.
Discussion	Returns an IBXCardInfo object.

createContainer()

Creates a BXContainer object.

Syntax	`public static IBXContainer createContainer (String` *key*`) throws Exception;`
Parameters	*key* — A string that contains the object mapping key in the format of `billerName.acctType`.
Discussion	This method is a class method. If you use only the default account, specify null for the billName acctType. Returns an IBXBillerContainer object.

createDocument()

Creates a document object.

Syntax	`public static IBXDocument createDocument (String` *key*`) throws Exception;`
Parameters	*key* — A string that contains the object mapping key in the format of `billerName.acctType`.
Discussion	The `createDocument` method is a class method. If you use only the default account, specify null for the billName acctType. Returns an IBXDetailDocument object.

createEventLog()

Creates an event log object.

Syntax	`public static IBXEventLog createEventLog (String` *key)* `throws Exception;`
Parameters	*key* — A string that contains the object mapping key in the format of `billerName.acctType`.
Discussion	The `createEventLog` method is a class method. If you use only the default account, specify null for the `billName` acctType. Returns an IBXEventLog object.

createMessage()

Creates a message object.

Syntax	`public static IBXMessage createMessage( String` *key* `) throws Exception;`
Parameters	*key* — A string that contains the object mapping key in the format of `billerName.acctType`.
Discussion	The `createMessage` method is a class method. If you use only the default account, specify null for the billName acctType. Returns an IBXMessage object.

createObject()

Creates an object.

Syntax	`public static IBXEntity createObject (String` *interfaceName*`, String` *key*`) throws Exception;`
Parameters	*interfaceName* — A string that specifies the class to which the object belongs. *key* — A string that contains the object mapping key in the format of a `billerName.acct.Type`.
Discussion	The `createObject` method is a protected class method. If you use only the default account, specify null for the billName acctType. Returns an IBXEntity object.

createPayHistory()

Creates a payment history object.

Syntax	`public static IBXPayHistory createPayHistory ( String` *key* `) throws Exception;`
Parameters	*key* — a string that contains the object mapping key in the format of a `billerName.acct.Type`.
Discussion	The `createPayHistory` is a class method. If you use only the default account, specify null for the billName acctType. Returns an IBXPayHistory object.

createPaySchedule()

Creates a payment schedule object.

Syntax	`public static IBXPaySchedule createPaySchedule (String` *key*`) throws Exception;`
Parameters	*key* — A string that contains the object mapping key in the format of a `billerName.acct.Type`.
Discussion	The `createPaySchedule` method is a class method. If you use only the default account, specify null for the billName acctType. Returns an IBXPaySchedule object.

createRequestLog()

Creates a request log object.

Syntax	`static IOFXRequestLog createRequestLog ( String` *key*`) throws Exception;`
Parameters	*key* — A string that contains the object mapping key
Discussion	Returns an IOFXRequestLog object.

createSummary()

Creates a summary object.

Contents	Description
Syntax	`public static IBXSummary createSummary (String` *key*`) throws Exception;`

Contents	Description
Parameters	*key* — A string that contains the object mapping key in the format of a `billerName.acct.Type`.
Discussion	The `createSummary` method is a class method. If you use only the default account, specify `null` for the *billName* parameter. Returns an IBXSummary object.

BXCardInfo Class Reference

The BXCardInfo class is the base class for credit card information.

Whenever possible, use the interface class IBXCardInfo (see "IBXCardInfo Class Reference) instead of BSPBillerInfo.

IBXCardInfo Class Reference

The IBXCardInfo class is the interface for credit card information.

Class Definition

Table 2-10 describes the IBXCardInfo class definition.

Table 2-10 IBXCardInfo Class Definition

Package	`netscape.billxpert.ejb.billing;`
Syntax	`public interface IBXCardInfo extends IBXBillingEntity;`

Methods

This section describes the IBXCardInfo class methods. Table 2-11 lists these methods.

Table 2-11 IBXCardInfo Methods *(1 of 3)*

Method	Description
`getType()`	Retrieves the card type (smart card or credit card).
`setType()`	Specifies the card type (smart card or credit card).
`getBrand()`	Retrieves the credit card type.

Table 2-11 IBXCardInfo Methods *(2 of 3)*

Method	Description
`setBrand()`	Specifies the credit card type
`getNumber()`	Retrieves the credit card number.
`setNumber()`	Specifies the credit card number.
`getDescription()`	Retrieves the credit card description.
`setDescription()`	Specifies the credit card description.
`getExpDate()`	Retrieves the credit card expiration date.
`setExpDate()`	Specifies the credit card expiration date.
`getOwner()`	Retrieves the credit card owner's name.
`setOwner()`	Specifies the credit card owner's name.
`getBillingName()`	Retrieves the credit card billing name.
`setBillingName()`	Specifies the credit card billing name.
`getAddress()`	Retrieves the credit card address.
`setAddress()`	Specifies the credit card address.
`getCity()`	Retrieves the credit card address city.
`setCity()`	Specifies the credit card address city.
`getState()`	Retrieves the credit card address state.
`setState()`	Specifies the credit card address state.
`getCountry()`	Retrieves the credit card address country.
`setCountry()`	Specifies the credit card address country.
`getPostalCode()`	Retrieves the credit card address postal code.
`setPostalCode()`	Specifies the credit card address postal code.
`getCardStatus()`	Retrieves the credit card address status.
`setCardStatus()`	Specifies the credit card address status.
`getPin()`	Retrieves the credit card address pin number.
`setPin()`	Specifies the credit card address pin number.
`getExtDate1()`	Retrieves the first extension date
`setExtDate1()`	Specifies the first extension date.
`getExtDate2()`	Retrieves the second extension date
`setExtDate2()`	Specifies the second extension date.

Table 2-11 IBXCardInfo Methods *(3 of 3)*

Method	Description
`getExtDate3()`	Retrieves the third extension date
`setExtDate3()`	Specifies the third extension date.
`getExtDate4()`	Retrieves the fourth extension date
`setExtDate4()`	Specifies the fourth extension date.
`getExtNum1()`	Retrieves the first extension number.
`setExtNum1()`	Specifies the first extension number.
`getExtNum2()`	Retrieves the second extension number.
`setExtNum2()`	Specifies the second extension number.
`getExtNum3()`	Retrieves the third extension number.
`setExtNum3()`	Specifies the third extension number.
`getExtNum4()`	Retrieves the fourth extension number.
`setExtNum4()`	Specifies the fourth extension number.

getType()

Retrieves the credit card or smart card type.

Syntax `public String getType();`

Parameters None.

Discussion

setType()

Specifies the credit card or smart card type.

Syntax `public void setType (String` *value*`);`

Parameters `value` — The credit card or smart card type.

Discussion

getBrand()

Retrieves the credit card type (Visa, MasterCard, AmEx, and so forth).

Syntax `public String getBrand();`

Parameters None.

Discussion

setBrand()

Retrieves the credit card type (Visa, MasterCard, Amex, and so forth).

Syntax `public void setBrand (String value);`

Parameters *value* — The credit card type.

Discussion

getNumber()

Retrieves the credit card number.

Syntax `public String getNumber();`

Parameters None.

Discussion

setNumber()

Specifies the credit card number.

Syntax `public String setNumber (String value);`

Parameters *value* — The credit card number.

Discussion

getDescription()

Retrieves the credit card description.

Syntax `public String getDescription();`

Parameters None.

Discussion

setDescription()

Specifies the credit card description.

Syntax `public void setDescription (String value);`

Parameters *value* — The credit card description.

Discussion

getExpDate()

Retrieves the credit card expiration date.

Syntax `public String getExpDate();`

Parameters None.

Discussion

setExpDate()

Specifies the credit card expiration date.

Syntax `public void setExpDate (String value);`

Parameters *value* — The credit card expiration date.

Discussion

getOwner()

Retrieves the credit card owner's name.

Syntax `public String getOwner();`

Parameters None.

Discussion

setOwner()

Specifies the credit card owner's name.

Syntax `public void setOwner (String value);`

Parameters *value* — The credit card owner's name.

Discussion

getBillingName()

Retrieves the credit card billing name.

Syntax `public String getBillingName();`

Parameters None.

Discussion

setBillingName()

Specifies the credit card billing name.

Syntax `public void setBillingName (String value);`

Parameters *value* — The billing name for the credit card.

Discussion

getAddress()

Retrieves the credit card address.

Syntax `public String getAddress();`

Parameters None.

Discussion

setAddress()

Specifies the credit card address.

Syntax `public void setBillingName (String value);`

Parameters *value* — The credit card address.

Discussion

getCity()

Retrieves the credit card address city name.

Syntax `public String getCity();`

Parameters None.

Discussion

setCity()

Specifies the credit card address city name.

Syntax `public void setCity (String value);`

Parameters *value* — The credit card address city name.

Discussion

getState()

Retrieves the credit card address state name.

Syntax `public void getState();`

Parameters None.

Discussion

setState()

Specifies the credit card address state name.

Syntax	`public void setState (String value);`
Parameters	*value* — The credit card address state name.
Discussion	

getCountry()

Retrieves the credit card address country name.

Syntax	`public String setCountry();`
Parameters	None.
Discussion	

setCountry()

Specifies the credit card address country name.

Syntax	`public void setCountry (String value);`
Parameters	*value* — The credit card address country name.
Discussion	

getPostalCode()

Retrieves the credit card address postal code.

Syntax	`public String getPostalCode();`
Parameters	None.
Discussion	

setPostalCode()

Specifies the credit card address postal code.

Syntax	`public void setPostalCode (String value);`
Parameters	*value* — The credit card address postal code.

Discussion

getCardStatus()

Retrieves the credit card status.

Syntax `public String getCardStatus();`

Parameters None.

Discussion

setCardStatus()

Retrieves the credit card status.

Syntax `public void setCardStatus (String` *value*`);`

Parameters *value* — The credit card status.

Discussion

getPin()

Retrieves the credit card pin number.

Syntax `public String getPin();`

Parameters None.

Discussion

setPin()

Specifies the credit card pin number.

Syntax `public void setPin (String` *value*`);`

Parameters *value* — The credit card pin number.

Discussion

getExtDate1()

Retrieves the first extension date.

Syntax `public Date getExtDate1();`

Parameters None.

Discussion

setExtDate1()

Specifies the credit card first extension date.

Syntax `public void setExtDate1 (Date` *date*`);`

Parameters *date* — The first extension date.

Discussion

getExtDate2()

Retrieves the second extension date.

Syntax `public Date getExtDate2();`

Parameters None.

Discussion

setExtDate2()

Specifies the second extension date.

Syntax `public void setExtDate1 (Date` *date*`);`

Parameters *date* — The second extension date.

Discussion

getExtDate3()

Retrieves the third extension date.

Syntax `public Date getExtDate3();`

Parameters None.

Discussion

setExtDate1()

Specifies the third extension date.

Syntax `public void setExtDate3 (Date` *date*`);`

Parameters *date* — The third extension date.

Discussion

getExtDate4()

Retrieves the fourth extension date.

Syntax `public Date getExtDate4();`

Parameters None.

Discussion

setExtDate4()

Specifies the fourth extension date.

Syntax `public void setExtDate4 (Date` *date*`);`

Parameters *date* — The fourth extension date.

Discussion

getExtNum1()

Retrieves the first extension number.

Syntax `public double getExtNum1();`

Parameters None.

Discussion

setExtNum1()

Specifies the first extension number.

Syntax `public void setExtNum1 (double num);`

Parameters *num* — The first extension number.

Discussion

getExtNum2()

Retrieves the second extension number.

Syntax `public double getExtNum2();`

Parameters None.

Discussion

setExtNum2()

Specifies the second extension number.

Syntax `public void setExtNum2 (double num);`

Parameters *num* — The second extension number.

Discussion

getExtNum3()

Retrieves the third extension number.

Syntax `public double getExtNum3();`

Parameters None.

Discussion

setExtNum3()

Specifies the third extension number.

Syntax `public void setExtNum3 (double num);`

Parameters	*num* — The third extension number.
Discussion	

getExtNum4()

Retrieves the fourth extension number.

Syntax	`public double getExtNum4();`
Parameters	None.
Discussion	

setExtNum4()

Specifies the fourth extension number.

Syntax	`public void setExtNum4 (double num);`
Parameters	*num* — The fourth extension number.
Discussion	

BXDocument Class Reference

The BXDocument class is the base class for defining methods that manipulate objects representing billing records.

Whenever possible, use the interface class IBXDocument (see “IBXDocument Class Reference) instead of BXDocument.

IBXDocument Class Reference

The `IBXDocument` interface class defines methods that manipulate objects representing document bills. You call the `BXBillingMgr.createDocument()` method to create a document object.

Before you can call the `IBXEntity.ejbCreate()` method (see “IBXBillingEntity Class Reference,” on page 83) to create a new record associated with the object, you must call the `setAccountID()` method to set the account ID.

Class Definition

Table 2-12 describes the IBXDocument interface class definition.

Table 2-12 BXDocument Class Definition

Package	`netscape.billxpert.ejb.billing;`
Syntax	`public interface IBXDocument extends IBXBillingEntity {...}`

Methods

This section describes the IBXDocument methods. Table 2-13 lists these methods.

Table 2-13 IBXDocument Class Methods *(1 of 3)*

Methods	Description
Manipulating billing records	
`ejbFindAllDetailsByType()`	Retrieves detail records of the specified type associated with this bill.
`ejbFindAllPaymentHistories()`	Retrieves payment history records associated with this bill.
`getSummary()`	Retrieves the summary associated with this bill.
Getting and setting billing information	
`getContainerID()`	Retrieves the bill's container ID.
`setContainerID()`	Specifies the bill's container ID.
`getAccountID()`	Retrieves the bill's account ID.
`setAccountID()`	Specifies the bill's account ID.
`getFileID()`	Retrieves the file ID.
`setFileID()`	Specifies the file ID.
`getDescription()`	Retrieves the bill's description.
`setDescription()`	Specifies the bill's description.
`getDocType()`	Retrieves the bill type.
`setDocType()`	Specifies the bill type.
`getBillType()`	Retrieves the bill's bill type.
`setBillType()`	Specifies the bill's bill type.
`getCreateDate()`	Retrieves the bill's creation date.
`getModifyDate()`	Retrieves the bill's modification date.

Table 2-13 IBXDocument Class Methods *(2 of 3)*

Methods	Description
`getFirstViewDate()`	Retrieves the bill's first view date.
`setFirstViewDate()`	Specifies the bill's first view date.
`getLastViewDate()`	Retrieves the bill's last view date.
`setLastViewDate()`	Specifies the bill's last view date.
`getViewCount()`	Retrieves the bill's view count.
`setViewCount()`	Specifies the bill's view count.
`getDocStatus()`	Retrieves the bill's status.
`setDocStatus()`	Specifies the bill's status.
`getCopyType()`	Retrieves the bill's copy type.
`setCopyType()`	Specifies the bill's copy type.
`getExtension()`	Retrieves the bill's extension.
`setExtension()`	Specifies the bill's extension.
`getBillDate()`	Retrieves the bill's bill date.
`setBillDate()`	Specifies the bill's bill date ID.
`getCanView()`	Determines if the bill is viewable
`setCanView()`	Specifies if the bill is viewable
`getInvoiceNum()`	Retrieves the invoice number.
`setInvoiceNum()`	Specifies the invoice number.
`incViewCount()`	Updates the view count and the last view date.
`getPMPrcStatus()`	Retrieves the price status.
`setPMPrcStatus()`	Specifies the price status.
`getPoNum()`	Retrieves the purchase order number.
`setPoNum()`	Specifies the purchase order number.
`getDeptNum()`	Retrieves the department order number.
`setDeptNum()`	Specifies the department order number.
`getExtDate1()`	Retrieves the first extension date
`setExtDate1()`	Specifies the first extension date.
`getExtDate2()`	Retrieves the second extension date
`setExtDate2()`	Specifies the second extension date.

Table 2-13 IBXDocument Class Methods *(3 of 3)*

Methods	Description
`getExtDate3()`	Retrieves the third extension date
`setExtDate3()`	Specifies the third extension date.
`getExtDate4()`	Retrieves the fourth extension date
`setExtDate4()`	Specifies the fourth extension date.
`getExtNum1()`	Retrieves the first extension number.
`setExtNum1()`	Specifies the first extension number.
`getExtNum2()`	Retrieves the second extension number.
`setExtNum2()`	Specifies the second extension number.
`getExtNum3()`	Retrieves the third extension number.
`setExtNum3()`	Specifies the third extension number.
`getExtNum4()`	Retrieves the fourth extension number.
`setExtNum4()`	Specifies the fourth extension number.

ejbFindAllDetailsByType()

Retrieves detail records of the specified type associated with this bill.

Syntax

```
public Vector ejbFindAllDetailsByType( String type, BXFilter filter,
BXOrder order) throws Exception;
```

Parameters

type — A string that specifies a detail record's type.

filter — A BXFilter object that specifies the Where clause of an SQL Select statement. If null, all records will be retrieved.

order — A BXOrder object that specifies the Order By clause of an SQL Select statement. If null, the retrieved records will be sorted in ascending order by their primary key.

Discussion

Retrieves all detail records associated with the bill that match type and clause in the *filter* parameter, ordered by the clause specified in the order parameter. For each record, creates an object, specifies the fields in the object from the record, and adds the object to the return vector.

You can specify the following types:

TelCoCall — Retrieves call detail records for IBXDetailCall

TelCoMisc — Retrieves detail records for IBXDetailMisc

TelCoRating — Retrieves rating detail records for IBXDetailRating

The bill object's database context must previously have been set by calling each object's `createBillingContext()` method. The account ID must be set in the bill object. Returns a vector that contains objects representing the selected records.

ejbFindAllPaymentHistories()

Retrieves payment history records associated with this bill, with the most recent ones first.

Syntax

```
public Vector ejbFindAllPaymentHistories ( BXFilter filter,
BXOrder order ) throws Exception;
```

Parameters

filter — If null, all records will be retrieved.

order — If null, the retrieved records will be sorted in ascending order by their primary key.

Discussion

Retrieves all payment history records associated with the bill that match the specified clause in the `filter` parameter, ordered by the clause specified in the `order` parameter. For each record, creates an object, specifies the fields in the object from the record, and adds the object to the return vector.

The bill object's database context must previously have been set by calling each object's `createBillingContext()` method. The account ID must be set in the bill object. Returns a vector that contains IBXPayHistory objects representing the selected records.

getAccountID()

Retrieves the bill's account ID.

Syntax

```
public String getAccountID();
```

Parameters	None.
Discussion	Returns a string that contains the account ID.

getBillDate()

Retrieves the bill's bill date.

Syntax	`public Date getBillDate ();`
Parameters	None.
Returns	Returns a Date object that contains the date.

getBillType()

Retrieves the bill's bill type.

Syntax	`public String getBillType ();`
Parameters	None.
Discussion	Returns a string that contains the type.

getCanView()

Determines if the bill is viewable.

Syntax	`public String getCanView();`
Parameters	None.
Discussion	Returns a string that determines if the bill is viewable.

getContainerID()

Retrieves the bill's container ID.

Syntax	`public String getContainerID();`
Parameters	None.
Discussion	Returns a string that contains the container ID.

getCopyType()

Retrieves the bill's copy type.

Syntax	`public String getCopyType();`
Parameters	None.
Discussion	Returns a string that contains the type.

getCreateDate()

Retrieves the bill's creation date.

Syntax	`public Date getCreateDate();`
Parameters	None.
Discussion	Returns a Date object that contains the date.

getDescription()

Retrieves the bill's description.

Syntax	`public String getDescription();`
Parameters	None.
Discussion	Returns a string that contains the description.

getDocStatus()

Retrieves the bill's status.

Syntax	`public String getDocStatus();`
Parameters	None.
Discussion	Returns a string that contains the status.

getDocType()

Retrieves the bill type.

Syntax	`public String getDocType();`

Parameters	None.
Discussion	Returns a string that contains the bill type.

getExtension()

Retrieves the bill's extension.

Syntax	`public String getExtension();`
Parameters	None.
Discussion	Returns a string that contains the extension ID.

getFileID()

Retrieves the file ID.

Syntax	`public String getFileID();`
Parameters	None.
Discussion	Returns a string that contains the ID.

getFirstViewDate()

Retrieves the bill's first view date.

Syntax	`public Date getFirstViewDate();`
Parameters	None.
Discussion	Returns a date object that contains the date.

getLastViewDate()

Retrieves the bill's last view date.

Syntax	`public Date getLastViewDate();`
Parameters	None.
Discussion	Returns a Date object that contains the date.

getModifyDate()

Retrieves the bill's modification date.

Syntax	`public Date getModifyDate();`
Parameters	None.
Discussion	Returns a Date object that contains the modification date.

getSummary()

Retrieves the summary associated with this bill.

Syntax	`public IBXSummary getSummary();`
Parameters	None.
Discussion	Returns an IBXSummary object that contains the summary.

getViewCount()

Retrieves the bill's view count.

Syntax	`public int getViewCount();`
Parameters	None.
Discussion	Returns an integer that contains the count.

setAccountID()

Specifies the bill's account ID.

Syntax	`public void setAccountID (String` *accountID*`);`
Parameters	*accountID* — A string that specifies the ID
Discussion	Only changes the value of the object field; it does not modify the database value.

setBillDate()

Specifies the bill's bill date.

Syntax	`public void setBillDate (Date` *date*`);`
Parameters	*date* — A Date object that specifies the date
Discussion	Only changes the value of the bill date field in the object; it does not modify the value in the database.

setBillType()

Specifies the bill type.

Syntax	`public void setBillType ( String` *billType* `);`
Parameters	*billType* — A string that specifies the bill type.
Discussion	Only changes the value of the object field; it does not modify the database value.

setCanView()

Specifies if the bill is viewable.

Syntax	`public void setCanView (String` *canView*`);`
Parameters	*canView* — A string that specifies if the bill is viewable.
Discussion	Only changes the value of the object field; it does not modify the database value.

setDocStatus()

Specifies the bill's status

Syntax	`public void setDocStatus( String` *status*`);`
Parameters	*status* — A string that specifies the status
Discussion	Only changes the value of the status field in the object; it does not modify the value in the database.

setDocType()

Specifies the bill type.

Syntax	`public void setDocType( String` *docType* `);`
Parameters	*docType* — A string that specifies the doc type
Discussion	Only changes the value of the doc type field in the object; it does not modify the value in the database.

setExtension()

Specifies the bill's extension.

Syntax	`public void setExtension( String` *extension*`);`
Parameters	*extension* — A string that specifies the extension
Discussion	Only changes the value of the extension field in the object; it does not modify the value in the database.

setFileID()

Specifies the file ID.

Syntax	`public void setFileID( String` *fileID*`);`
Parameters	*fileID* — A string that specifies the file ID
Discussion	Only changes the value of the file ID field in the object; it does not modify the value in the database.

setFirstViewDate()

Specifies the bill's first view date.

Syntax	`public void setFirstViewDate( Date` *date* `);`
Parameters	*date* — A string that specifies the date
Discussion	Only changes the value of the object field; it does not modify the database value.

setLastViewDate()

Specifies the bill's last view date.

Syntax	`public void setLastViewDate( Date` *`date`* `);`
Parameters	*date* — A string that specifies the date
Discussion	Only changes the value of the object field; it does not modify the database value.

setViewCount()

Specifies the bill's view count.

Syntax	`public void setViewCount (int` *`viewCount`*`);`
Parameters	*viewCount* — A string that specifies the view count
Discussion	Only changes the value of the view count field in the object; it does not modify the value in the database.

getExtension()

Retrieves the detail line's extension.

Syntax	`public String getExtension();`
Parameters	None.
Discussion	Returns a string that contains the Extension.

getFileID()

Retrieves the file ID.

Syntax	`public String getFileID();`
Parameters	None.
Discussion	Returns a string that contains the ID.

setDescription()

Specifies the detail line's description.

Syntax	`public void setDescription (String` *description*`);`
Parameters	*description* — A string that specifies the description.
Discussion	Only changes the value of the object field; it does not modify the database value.

setFileID()

Specifies the file ID.

Syntax	`public void setFileID (String` *fileID*`);`
Parameters	*fileID* — A string that specifies the file ID
Discussion	Only changes the value of the field in the object; it does not modify the value in the database.

getInvoiceNum()

Retrieves the invoice number.

Syntax	`public String getInvoiceNum();`
Parameters	None.
Discussion	Only changes the value of the field in the object; it does not modify the value in the database.

setInvoiceNum()

Specifies the invoice number.

Syntax	`public void setInvoiceNum (String` *invoiceNum*`);`
Parameters	*invoiceNum* — A string that specifies the invoice number.
Discussion	Only changes the value of the field in the object; it does not modify the value in the database.

incViewCount()

Updates the view count and the last view date.

Syntax	`public void incViewCount() throws Exception;`
Parameters	None.
Discussion	Only changes the value of the field in the object; it does not modify the value in the database.

getPMPrcStatus()

Retrieves the price status.

Syntax	`public String getPMPrcStatus();`
Parameters	None.
Discussion	Only changes the value of the field in the object; it does not modify the value in the database.

setPMPrcStatus()

Specifies the price status.

Syntax	`public void setPMPrcStatus (String` *pmprcStatus*`);`
Parameters	*pmprcStatus* — A string that specifies the price status.
Discussion	Only changes the value of the field in the object; it does not modify the value in the database.

getPoNum()

Retrieves the purchase order number.

Syntax	`public String getPMPrcStatus();`
Parameters	None.
Discussion	Only changes the value of the field in the object; it does not modify the value in the database.

setPoNum()

Specifies the purchase order number.

Syntax	`public void setPMPrcStatus (String` *poNum*`);`
Parameters	*poNum* — A string that specifies the purchase order number.
Discussion	Only changes the value of the field in the object; it does not modify the value in the database.

getDeptNum()

Retrieves the department number.

Syntax	`public String getDeptNum();`
Parameters	None.
Discussion	Only changes the value of the field in the object; it does not modify the value in the database.

setDeptNum()

Specifies the department number.

Syntax	`public void setDeptNum (String` *deptNum*`);`
Parameters	*deptNum* — A string that specifies the department number.
Discussion	Only changes the value of the field in the object; it does not modify the value in the database.

getExtDate1()

Retrieves the first extension date.

Syntax	`public Date getExtDate1();`
Parameters	None.
Discussion	

setExtDate1()

Specifies the first extension date.

Syntax `public void setExtDate1 (Date date);`

Parameters *date* — A date that indicates the first extension date.

Discussion

getExtDate2()

Retrieves the second extension date.

Syntax `public Date getExtDate2();`

Parameters None.

Discussion

setExtDate2()

Specifies the second extension date.

Syntax `public void setExtDate2 (Date date);`

Parameters *date* — A date that indicates the second extension date.

Discussion

getExtDate3()

Retrieves the third extension date.

Syntax `public Date getExtDate3();`

Parameters None.

Discussion

setExtDate3()

Specifies the third extension date.

Syntax `public void setExtDate3 (Date date);`

Parameters `date` — A date that indicates the third extension date.

Discussion

getExtDate4()

Retrieves the fourth extension date.

Syntax
```
public Date getExtDate4();
```

Parameters None.

Discussion

setExtDate4()

Specifies the fourth extension date.

Syntax
```
public void setExtDate4 (Date date);
```

Parameters *date* — A date that indicates the fourth extension date.

Discussion

getExtNum1()

Retrieves the first extension number.

Syntax
```
public double getExtNum1();
```

Parameters None.

Discussion

setExtNum1()

Specifies the first extension number.

Syntax
```
public void setExtNum1 (double num);
```

Parameters *num* — A double that indicates the first extension number.

Discussion

getExtNum2()

Retrieves the second extension number.

Syntax `public double getExtNum2();`

Parameters None.

Discussion

setExtNum2()

Specifies the second extension number.

Syntax `public void setExtNum1 (double num);`

Parameters *num* — A double that indicates the second extension number.

Discussion

getExtNum3()

Retrieves the third extension number.

Syntax `public double getExtNum3();`

Parameters None.

Discussion

setExtNum3()

Specifies the third extension number.

Syntax `public void setExtNum1 (double num);`

Parameters *num* — A double that indicates the third extension number.

Discussion

getExtNum4()

Retrieves the fourth extension number.

Syntax `public double getExtNum4();`

Parameters None.

Discussion

setExtNum4()

Specifies the fourth extension number.

Syntax `public void setExtNum1 (double num);`

Parameters *num* — A double that indicates the fourth extension number.

Discussion

setExtDate4()

Specifies the fourth extension date.

Syntax `public void setExtDate4 (Date date);`

Parameters `date` — A date that indicates the fourth extension date.

Discussion

getExtNum1()

Retrieves the first extension number.

Syntax `public double getExtNum1();`

Parameters None.

Discussion

setExtNum1()

Specifies the first extension number.

Syntax `public void setExtNum1 (double num);`

Parameters *num* — A double that indicates the first extension number.

Discussion

getExtNum2()

Retrieves the second extension number.

Syntax `public double getExtNum2();`

Parameters None.

Discussion

setExtNum2()

Specifies the second extension number.

Syntax `public void setExtNum1 (double num);`

Parameters *num* — A double that indicates the second extension number.

Discussion

getExtNum3()

Retrieves the third extension number.

Syntax `public double getExtNum3();`

Parameters None.

Discussion

setExtNum3()

Specifies the third extension number.

Syntax `public void setExtNum1 (double num);`

Parameters *num* — A double that indicates the third extension number.

Discussion

getExtNum4()

Retrieves the fourth extension number.

Syntax `public double getExtNum4();`

Parameters	None.
Discussion	

setExtNum4()

Specifies the fourth extension number.

BXSummary Class Reference

The BXSummary base class defines methods that manipulate objects representing billing summary records.

Whenever possible, use the interface class IBXSummary (see “IBXSummary Class Reference) instead of BXSummary.

IBXSummary Class Reference

The IBXSummary interface class defines methods that manipulate objects representing billing summary records. You call the `BXBillingMgr.createSummary()` method to create Returns a summary object.

Before you can call the `IBXEntity.ejbCreate()` (see “IBXBillingEntity Class Reference,” on page 83) method to create a new record associated with the object, you must call the `setdocID()`, `setStartDate()`, `setEndDate()`, and `setDueDate()` methods to set the document ID, and summary start, end, and due dates, respectively.

Class Definition

Table 2-14 describes the IBXSummary class interface definition.

Table 2-14 IBXSummary Class Definition

Package	`netscape.billxpert.ejb.billing`
Syntax	`public interface IBXSummary extends IBXBillingEntity {...}`

Methods

This section describes the IBXSummary class methods. Table 2-15 lists these methods.

Table 2-15 IBXSummary Class Methods *(1 of 3)*

Methods	Description
Manipulating summary-related records	
`ejbFindByDocumentID()`	Retrieves the summary associated with this document.
`ejbFindByContainerID()`	Retrieves the summary associated with this container.
Getting and setting summary information	
`getContainerID()`	Retrieves the summary's container ID.
`setContainerID()`	Specifies the summary's container ID.
`getBillerID()`	Retrieves the Biller ID.
`setBillerID()`	Specifies the Biller ID.
`getDocID()`	Retrieves the summary's document ID.
`setDocID()`	Specifies the summary's document ID.
`getFileID()`	Retrieves the file ID.
`setFileID()`	Specifies the file ID.
`getAutoPayCalculated()`	Retrieves the autopay calculation.
`setAutoPayCalculated()`	Specifies the autopay calculation.
`getStartDate()`	Retrieves the starting date.
`setStartDate()`	Specifies the starting date.
`getEndDate()`	Retrieves the ending date.
`setEndDate()`	Specifies the ending date.
`getOriginalAmount()`	Retrieves the original amount.
`setOriginalAmount()`	Specifies the original amount.
`getCurrentBalance()`	Retrieves the current balance.
`setCurrentBalance()`	Specifies the current balance.
`getCurrentCharge()`	Retrieves the current charge.
`setCurrentCharge()`	Specifies the current charge.
`getPreviousBalance()`	Retrieves the previous balance.
`setPreviousBalance()`	Specifies the previous balance.

Table 2-15 IBXSummary Class Methods *(2 of 3)*

Methods	Description
`getPreviousPayment()`	Retrieves the previous payment.
`setPreviousPayment()`	Specifies the previous payment.
`getBalanceDate()`	Retrieves the balance date.
`setBalanceDate()`	Specifies the balance date.
`getDueDate()`	Retrieves the due date.
`setDueDate()`	Specifies the due date.
`getLateAfterDate()`	Retrieves the late-after date.
`setLateAfterDate()`	Specifies the late-after date.
`getPenaltyType()`	Retrieves the penalty type.
`setPenaltyType()`	Specifies the penalty type.
`getPenaltyRate()`	Retrieves the penalty rate.
`setPenaltyRate()`	Specifies the penalty rate.
`getMinPayment()`	Retrieves the minimum payment.
`setMinPayment()`	Specifies the minimum payment.
`getPaymentAmount()`	Retrieves the payment amount.
`setPaymentAmount()`	Specifies the payment amount.
`getLastPayDate()`	Retrieves the last payment date.
`setLastPayDate()`	Specifies the last payment date.
`getPayStatus()`	Retrieves the payment status.
`setPayStatus()`	Specifies the payment status.
`getSumStatus()`	Retrieves the summary status.
`setSumStatus()`	Specifies the summary status.
`getPayterms()`	Retrieves the payment terms code.
`setPayterms()`	Specifies the payment terms code.
`getDiscountBeforeDate()`	Retrieves the discount-before date.
`setDiscountBeforeDate()`	Specifies the discount-before date.
`getDiscountType()`	Retrieves the discount type.
`setDiscountType()`	Specifies the discount type.
`getExtDate1`	Retrieves the first extension date

Table 2-15 IBXSummary Class Methods *(3 of 3)*

Methods	Description
`setExtDate1`	Specifies the first extension date.
`getExtDate2()`	Retrieves the second extension date
`setExtDate2()`	Specifies the second extension date.
`getExtDate3()`	Retrieves the third extension date
`setExtDate3()`	Specifies the third extension date.
`getExtDate4()`	Retrieves the fourth extension date
`setExtDate4()`	Specifies the fourth extension date.
`getExtNum1()`	Retrieves the first extension number.
`setExtNum1()`	Specifies the first extension number.
`getExtNum2()`	Retrieves the second extension number.
`setExtNum2()`	Specifies the second extension number.
`getExtNum3()`	Retrieves the third extension number.
`setExtNum3()`	Specifies the third extension number.
`getExtNum4()`	Retrieves the fourth extension number.
`setExtNum4()`	Specifies the fourth extension number.

ejbFindByContainerID()

Retrieves the summary associated with this container.

Syntax `public void ejbFindByContainerID ( String` *key* `) throws Exception;`

Parameters *key* — A string that specifies the container ID.

Discussion The *key* parameter specifies the record to read. This method throws an exception if the record with this key does not exist. The fields of this object are set by from the contents of the record. The database context must have been set by calling the `createBillingContext()` method.

ejbFindByDocumentID()

Retrieves the summary associated with this document.

Syntax `public void ejbFindByDocumentID ( String` *key* `) throws Exception;`

Parameters	*key* — A string that specifies the document ID
Discussion	The `key` parameter specifies the record to read. This method throws an exception if the record with this key does not exist. The fields of this object are set by from the contents of the record. The database context must have been set by calling the `createBillingContext()` method.

getBalanceDate()

Retrieves the balance date.

Syntax	`public Date getBalanceDate();`
Parameters	None.
Discussion	Returns a Date object that contains the balance date.

getContainerID()

Retrieves the summary's container ID.

Syntax	`public String getContainerID();`
Parameters	None.
Discussion	Returns a string that contains the ID.

getBillerID()

Retrieves the biller identification.

Syntax	`public String getBillerID();`
Parameters	None.
Discussion	Returns a string that contains the Biller ID.

getCurrentBalance()

Retrieves the current balance.

Syntax	`public double getCurrentBalance();`
Parameters	None.

Discussion	Returns a double-precision floating point number that contains the balance.

getCurrentCharge()

Retrieves the current charge.

Syntax	`public double getCurrentCharge();`
Parameters	None.
Discussion	Returns a double-precision floating point number that contains the amount.

getDiscountBeforeDate()

Retrieves the discount-before date.

Syntax	`public Date getDiscountBeforeDate();`
Parameters	None.
Discussion	Returns a Date object that contains the date.

getDescription()

Retrieves the summary's description.

Syntax	`public String getDescription();`
Parameters	None.
Discussion	Returns a string that contains the description.

getDiscountRate()

Retrieves the discount rate.

Syntax	`public double getDiscountRate();`
Parameters	None.
Discussion	Returns a double-precision floating point number that contains the rate.

getDiscountType()

Retrieves the discount type.

Syntax	`public String getDiscountType();`
Parameters	None.
Discussion	Returns a string that contains the discount type.

getDocID()

Retrieves the summary's document ID.

Syntax	`public String getDocID;`
Parameters	None.
Discussion	Returns a string that contains the document ID.

getDueDate()

Retrieves the due date.

Syntax	`public Date getDueDate();`
Parameters	None.
Discussion	Returns a Date object that contains the due date.

getEndDate()

Retrieves the ending date.

Syntax	`public Date getEndDate();`
Parameters	None.
Discussion	Returns a Date object that contains the date.

getExtension()

Retrieves the summary's extension.

Syntax	`public String getExtension();`

Parameters	None.
Discussion	Returns a string that contains the extension.

getFileID()

Retrieves the file ID.

Syntax	`public String getFileID();`
Parameters	None.
Discussion	Returns a string that contains the file ID.

getAutoPayCalculated()

Retrieves the autopay calculation.

Syntax	`public String getAutoPayCalculated();`
Parameters	None.
Discussion	Returns a string that contains the autopay calculation.

getLastPayDate()

Retrieves the last payment date.

Syntax	`public Date getLastPayDate();`
Parameters	None.
Discussion	Returns a Date object that contains the last payment date.

getLateAfterDate()

Retrieves the date after which the payment is late.

Syntax	`public Date getLateAfterDate();`
Parameters	None.
Discussion	Returns a Date object that contains the date after which the payment is late.

getMinPayment()

Retrieves the minimum payment.

Syntax	`public double getMinPayment();`
Parameters	None.
Discussion	Returns a double-precision floating point number that contains the minimum payment amount.

getOriginalAmount()

Retrieves the original amount.

Syntax	`public double getOriginalAmount();`
Parameters	None.
Discussion	Returns a double-precision floating point number that contains the original amount.

getPayStatus()

Retrieves the payment status.

Syntax	`public String getPayStatus();`
Parameters	None.
Discussion	Returns a string that contains the payment status.

getPaymentAmount()

Retrieves the payment amount.

Syntax	`public double getPaymentAmount();`
Parameters	None.
Discussion	Returns a double-precision floating point number that contains the payment amount.

getPayterms()

Retrieves the payment terms code.

Syntax	`public String getPayterms();`
Parameters	None.
Discussion	Returns a string that contains the code.

getPenaltyRate()

Retrieves the penalty rate.

Syntax	`public double getPenaltyRate();`
Parameters	None.
Discussion	Returns a double-precision floating point number that contains the penalty rate.

getPenaltyType()

Retrieves the penalty type.

Syntax	`public String getPenaltyType();`
Parameters	None.
Discussion	Returns a string that contains the penalty type.

getPreviousBalance()

Retrieves the previous balance.

Syntax	`public double getPreviousBalance();`
Parameters	None.
Discussion	Returns a double-precision floating point number that contains the previous balance.

getPreviousPayment()

Retrieves the previous payment.

Syntax	`public double getPreviousPayment();`
Parameters	None.
Discussion	Returns a double-precision floating point number that contains the previous payment amount.

getStartDate()

Retrieves the starting date.

Syntax	`public Date getStartDate();`
Parameters	None.
Discussion	Returns a Date object that contains the starting date.

getSumStatus()

Retrieves the summary status.

Syntax	`public String getSumStatus();`
Parameters	None.
Discussion	Returns a string that contains the summary status.

getURL()

Retrieves the URL.

Syntax	`public String getURL();`
Parameters	None.
Discussion	Returns a string that contains the URL.

setBalanceDate()

Specifies the balance date.

Syntax	`public void setBalanceDate( Date` *date* `);`
Parameters	*date* — A Date object that specifies the balance date.
Discussion	Only changes the value of the balance date in the object; it does not modify the value in the database.

setContainerID()

Specifies the summary's container ID.

Syntax	`public void setContainerID( String` *containerID*`);`
Parameters	*containerID* — A string that specifies the summary's container ID.
Discussion	Only changes the value of the container ID in the object; it does not modify the value in the database.

setBillerID()

Specifies the Biller ID.

Syntax	`public void setBillerID ( String` *billerID* `);`
Parameters	*billerID* — A string that specifies the biller ID
Discussion	Only changes the value of the billerID in the object; it does not modify the value in the database.

setCurrentBalance()

Specifies the current balance.

Syntax	`public void setCurrentBalance( double` *amount* `);`
Parameters	*amount* — A double-precision floating point number that specifies the amount
Discussion	Only changes the value of the current balance in the object; it does not modify the value in the database.

setCurrentCharge()

Specifies the current charge.

Syntax	`public void setCurrentCharge( double` *amount*`);`
Parameters	*amount* — A double-precision floating point number that specifies the amount
Discussion	Only changes the value of the current charge in the object; it does not modify the value in the database.

setDescription()

Specifies the summary's description.

Syntax	`public void setDescription( String` *description*`);`
Parameters	*description* — A string that specifies the description.
Discussion	Only changes the value of the description in the object; it does not modify the value in the database.

setDiscountBeforeDate()

Specifies the discount-before date.

Syntax	`public void setBalanceDate( Date` *date* `);`
Parameters	*date* — A Date object that specifies the date
Discussion	Only changes the value of the balance date in the object; it does not modify the value in the database.

setDiscountRate()

Specifies the discount rate.

Syntax	`public void setDiscountRate( double` *amount*`);`
Parameters	*amount* — A double-precision floating point number that specifies the amount
Discussion	Only changes the value of the amount in the object; it does not modify the value in the database.

setDiscountType()

Specifies the dispute reason.

Syntax	`public void setDiscountType( String` *type*`);`
Parameters	*type* — A string that specifies the type
Discussion	Only changes the value of the type in the object; it does not modify the value in the database.

setDocID()

Specifies the summary's document ID.

Syntax	`public void setDocID( String` *docID*`);`
Parameters	*docID* — A string that specifies the document ID.
Discussion	Only changes the value of the doc ID in the object; it does not modify the value in the database.

setDueDate()

Specifies the due date.

Syntax	`public void setDueDate( Date` *date*`);`
Parameters	*date* — A string that specifies the date.
Discussion	Only changes the value of the date in the object; it does not modify the value in the database.

setEndDate()

Specifies the ending date.

Syntax	`public void setEndDate (Date` *date*`);`
Parameters	*date* — A Date object that specifies the date.
Discussion	Only changes the value of the ending date in the object; it does not modify the value in the database.

setExtension()

Specifies the summary's extension.

Syntax	`public void setExtension( String` *extension*);
Parameters	*extension* — A string that specifies the extension
Discussion	Only changes the value of the extension field in the object; it does not modify the value in the database.

setFileID()

Specifies the file ID.

Syntax	`public void setFileID( String` *fileID*`);`
Parameters	*fileID* — A string that specifies the ID.
Discussion	Only changes the value of the file ID in the object; it does not modify the value in the database.

setAutoPayCalculated()

Specifies the automatic payment calculation.

Syntax	`public void setAutoPayCalculated ( String` *autoPayCalculated*`);`
Parameters	*autoPayCalculated* — A string that specifies the automatic pay calculation.
Discussion	Only changes the value of the automatic pay calculated in the object; it does not modify the value in the database.

setLastPayDate()

Specifies the last payment date.

Syntax	`public void setLastPayDate (Date` *date*`);`
Parameters	*date* — A date object that specifies the last payment date.
Discussion	Only changes the value of the date in the object; it does not modify the value in the database.

setLateAfterDate()

Specifies the date after which the payment is late.

Syntax	`public void setLateAfterDate (Date` *date*`);`
Parameters	*date* — A date object that specifies the date after which the payment is late.
Discussion	Only changes the value of the late date in the object; it does not modify the value in the database.

setMinPayment()

Specifies the minimum payment.

Syntax	`public void setMinPayment( double` *amount*`);`
Parameters	*amount* — A double-precision floating point number that specifies the minimum payment amount.
Discussion	Only changes the value of the amount in the object; it does not modify the value in the database.

setOriginalAmount()

Specifies the original amount.

Syntax	`public void setOriginalAmount( double` *amount*`);`
Parameters	*amount* — A double-precision floating point number that specifies the original amount.
Discussion	Only changes the value of the amount in the object; it does not modify the value in the database.

setPayStatus()

Specifies the payment status.

Syntax	`public void setPayStatus( String` *status*`);`
Parameters	*status* — A string that specifies the status.
Discussion	Only changes the value of the payment status in the object; it does not modify the value in the database.

setPaymentAmount()

Specifies the original amount.

Syntax	`public void setPaymentAmount( double` *amount*`);`
Parameters	*amount* — A double-precision floating point number that specifies the amount.
Discussion	Only changes the value of the amount in the object; it does not modify the value in the database.

setPayterms()

Specifies the payment terms code.

Syntax	`public void set PayTerms( String` *code*`);`
Parameters	*code* — A string that specifies the code.
Discussion	Only changes the value of the payment terms code in the object; it does not modify the value in the database.

setPenaltyRate()

Specifies the penalty rate.

Syntax	`public void setPenaltyRate( double` *amount*`);`
Parameters	*amount* — A double precision floating point number that specifies the penalty rate.
Discussion	Only changes the value of the penalty rate in the object; it does not modify the value in the database.

setPenaltyType()

Specifies the penalty type.

Syntax	`public void setPenaltyType( String` *penaltyType*`);`
Parameters	*penaltyType* — A string that specifies the penalty type.
Discussion	Only changes the value of the penalty type in the object; it does not modify the value in the database.

setPreviousBalance()

Displays previous balance.

Syntax	`public void setPreviousBalance( double` *amount*`);`
Parameters	*amount* — A double precision floating number that specifies the amount.
Discussion	Only changes the value of the previous payment in the object; it does not modify the value in the database.

setStartDate()

Specifies the starting date.

Syntax	`public void setStartDate( Date` *date*`);`
Parameters	*date* — A Date object that specifies the start date.
Discussion	Only changes the value of the start date in the object; it does not modify the value in the database.

setSumStatus()

Specifies the summary status.

Syntax	`public void setSumStatus( String` *status*`);`
Parameters	*status* — A string that specifies the summary status.
Discussion	Only changes the value of the summary status in the object; it does not modify the value in the database.

setURL()

Specifies the URL.

Syntax	`public void setURL( String` *url*`);`
Parameters	*url* — A string that specifies the URL.
Discussion	Only changes the value of the URL in the object; it does not modify the value in the database.

getExtDate1()

Retrieves the first extension date.

Syntax `public Date getExtDate1();`

Parameters None.

Discussion

setExtDate1()

Specifies the first extension date.

Syntax `public void setExtDate1(Date` *date*`);`

Parameters *date* — A date that indicates the first extension date.

Discussion

getExtDate2()

Retrieves the second extension date.

Syntax `public Date getExtDate2();`

Parameters None.

Discussion

setExtDate2()

Specifies the second extension date.

Syntax `public void setExtDate2 (Date` *date*`);`

Parameters *date* — A date that indicates the second extension date.

Discussion

getExtDate3()

Retrieves the third extension date.

Syntax `public Date getExtDate3();`

Parameters None.

Discussion

setExtDate3()

Specifies the third extension date.

Syntax `public void setExtDate3 (Date date);`

Parameters *date* — A date that indicates the third extension date.

Discussion

getExtDate4()

Retrieves the fourth extension date.

Syntax `public Date getExtDate4();`

Parameters None.

Discussion

setExtDate4()

Specifies the fourth extension date.

Syntax `public void setExtDate4 (Date date);`

Parameters `date` — A date that indicates the fourth extension date.

Discussion

getExtNum1()

Retrieves the first extension number.

Syntax `public double getExtNum1();`

Parameters None.

Discussion

setExtNum1()

Specifies the first extension number.

Syntax `public void setExtNum1 (double num);`

Parameters *num* — A double that indicates the first extension number.

Discussion

getExtNum2()

Retrieves the second extension number.

Syntax `public double getExtNum2();`

Parameters None.

Discussion

setExtNum2()

Specifies the second extension number.

Syntax `public void setExtNum1 (double num);`

Parameters *num* — A double that indicates the second extension number.

Discussion

getExtNum3()

Retrieves the third extension number.

Syntax `public double getExtNum3();`

Parameters None.

Discussion

setExtNum3()

Specifies the third extension number.

Syntax `public void setExtNum1 (double num);`

Parameters	*num* — A double that indicates the third extension number.
Discussion	

getExtNum4()

Retrieves the fourth extension number.

Syntax	`public double getExtNum4();`
Parameters	None.
Discussion	

setExtNum4()

Specifies the fourth extension number.

Syntax	`public void setExtNum1 (double num);`
Parameters	*num* — A double that indicates the fourth extension number.
Discussion	

Chapter 3

Membership Component Business Object Interfaces

This chapter discusses component business object interface classes for membership records. These classes provide access to records stored on a Directory Server database using the Lightweight Directory Access Protocol (LDAP).

This chapter includes the following sections:

- Overview
- Membership Class Reference Descriptions

Overview

Membership classes represent LDAP records on the Directory Server. You typically call a class method of the BXMembershipMgr class to create an object and call the object's `setMembershipContext()` method to associate it with a context.

Once the object has been created and associated with a context, you can use the Returns access methods described in "IBXEntity Class Reference" to create, retrieve, update, and delete a record in the Directory Server database.

Membership Classes

Membership classes represent LDAP records on the Directory Server. You use these classes to manipulate membership objects.Table 3-1 lists these classes.

Table 3-1 Membership Classes / Interfaces

Class Name	Description	Location
1. BXAddress	Implementation of IBXAddress.	page 146
2. IBXAddress	Interface that defines address objects.	page 147
3. BXBillerProfile	Implementation of IBXBillerProfile.	page 152
4. IBXBillerProfile	Interface that defines billing organization objects.	page 152
5. BXCustomerProfile	Implementation of IBXCustomerProfile.	page 176
6. IBXCustomerProfile	Interface that defines customer profile objects.	page 176
7. BXFinancialAccount	Implementation of IBXFinancialAccount.	page 202
8. IBXFinancialAccount	Manipulates financial account objects.	page 202
9. BXMembershipContext	Establishes a context between an AppLogic and a membership business object.	page 206
10. BXMembershipEntity	Base class that is subclassed by all membership classes.	page 211
11. IBXMembershipEntity	Base interface for membership business objects.	page 211
12. BXMembershipMgr	Creates membership-related business objects.	page 212
13. BXPaymentType	Implementation of IBXPaymentType.	page 216
14. IBXPaymentType	Interface that defines payment type objects.	page 217

BXAddress Class Reference

The BXAddress class is an implementation of IBXAddress interface, which defines address objects.

Whenever possible, use the interface class IBXAddress (see “IBXAddress Class Reference”) instead of BXAddress.

IBXAddress Class Reference

The IBXAddress interface defines address objects that represent customer address record objects. You call the `BXMembershipMgr.createAddress()` method to create an address object.

Class Definition

Table 3-2 describes the IBXAddress interface definition.

Table 3-2 IBXAddress Interface Definition

Package	`com.iplanet.ecomm.billxb2b.ejb.membership;`
Syntax	`public interface IBXAddress extends IBXMembershipEntity {...}`

Methods

This section describes the IBXAddress methods. Table 3-3 lists these methods.

Table 3-3 IBXAddress Class Methods

Method	Description
Getting and setting address information	
`getType()`	Determines the address type.
`setType()`	Specifies the address type.
`getAddress1()`	Determines the first line of the address.
`setAddress1()`	Specifies the first line of the address.
`getAddress2()`	Determines the second line of the address.
`setAddress2()`	Specifies the second line of the address.
`getAddress3()`	Determines the third line of the address.
`setAddress3()`	Specifies the third line of the address.
`getCity()`	Determines the city.
`setCity()`	Specifies the city.
`getState()`	Determines the state.
`setState()`	Specifies the state.
`getPostalCode()`	Determines the zip or postal code.

Table 3-3 IBXAddress Class Methods

Method	Description
`setPostalCode()`	Specifies the zip or postal code.
`getCountry()`	Determines the country.
`setCountry()`	Specifies the country.
`getAddressStatus()`	Determines the address status.
`setAddressStatus()`	Specifies the address status.

getAddress1()

Determines the first line of the address.

Syntax	`String getAddress();`
Parameters	None.
Discussion	Returns a string that contains the first line of the address.

getAddress2()

Determines the second line of the address.

Syntax	`String getAddress2();`
Parameters	None.
Discussion	Returns a string that contains the second line of the address.

getAddress3()

Determines the third line of the address.

Syntax	`String getAddress3();`
Parameters	None.
Discussion	Returns a string that contains the third line of the address.

getAddressStatus()

Determines the address status.

Syntax	`String getAddressStatus();`
Parameters	None.
Discussion	Returns a string that contains the address status.

getCity()

Determines the city.

Syntax	`String getCity();`
Parameters	None.
Discussion	Returns a string that contains the city of the address.

getCountry()

Determines the country.

Syntax	`String getCountry();`
Parameters	None.
Discussion	Returns a string that specifies the country of the address.

getPostalCode()

Determines the zip or postal code.

Syntax	`String getPostalCode();`
Parameters	None.
Discussion	Returns a string that specifies the zip code of the address.

getState()

Determines the state.

Syntax	`String getState();`
Parameters	None.
Discussion	Returns a string that specifies the state of the address.

getType()

Determines the address type.

Syntax	`String getType();`
Parameters	None.
Discussion	Returns a string that specifies the type of the address.

setAddress1()

Specifies the first line of the address.

Syntax	`void setAddress1 (String` *value*`);`
Parameters	*value* — A string that specifies the first line of the address.
Discussion	

setAddress2()

Specifies the second line of the address.

Syntax	`void setAddress2 (String` *value*`);`
Parameters	*value* — A string that specifies the second line of the address.
Discussion	

setAddress3()

Specifies the third line of the address.

Syntax `void setAddress3 (String value);`

Parameters *value* — A string that specifies the third line of the address.

setAddressStatus()

Specifies the address status.

Syntax `void setAddressStatus (String value);`

Parameters *value* — A string that specifies the address status.

Discussion

setCity()

Specifies the city.

Syntax `void setCity (String value);`

Parameters *value* — A string that specifies the city

Discussion

setCountry()

Specifies the country.

Syntax `void setCountry (String value);`

Parameters *value* — A string that specifies the country

Discussion

setPostalCode()

Specifies the postal code.

Syntax `void setPostalCode (String value);`

Parameters	*value* — A string that specifies the postal code.
Discussion	

setState()

Specifies the state.

Syntax	`void setState (String value);`
Parameters	*value* — A string that specifies the state.
Discussion	

setType()

Specifies the address type.

Syntax	`void setType (String value);`
Parameters	*value* — A string that specifies the address type.
Discussion	

BXBillerProfile Class Reference

The BXBillerProfile class is an implementation of IBXBillerProfile interface, which defines billing organization objects.

Whenever possible, use the interface class IBXBillerProfile (see “IBXBillerProfile Class Reference”) instead of BXBillerProfile.

IBXBillerProfile Class Reference

The IBXBillerProfile interface defines billing organization objects that represent biller profile records. You call the `BXMembershipMgr.createBillerProfile()` method to create a biller profile object.

Before you can call the `IBXEntity.ejbCreate()` method (see page 129) to create a new record Returns associated with the object, you must call the `setOrgName()` method to set the object’s account type field.

Interface Definition

Table 3-4 describes the IBXBillerProfile interface definition.

Table 3-4 IBXBillerProfile Interface Definition

Package	`com.iplanet.ecomm.billxb2b.ejb.membership;`
Syntax	`public interface IBXBillerProfile extends IBXMembershipEntity {...}`

Methods

This section describes the IBXBillerProfile class method. Table 3-5 lists these methods.

Table 3-5 IBXBillerProfile Class Methods *(1 of 5)*

Methods	Description
Retrieving biller-related records	
`ejbFindByOrganizationName()`	Retrieves the biller profile record specified by the organization name.
`ejbFindAllPersons()`	Retrieves the customer profile records associated with this biller.
`ejbFindAllCompanies ()`	Retrieves the companies belonging to this organization.
`ejbFindAllFinancialAccounts()`	Retrieves the financial account records associated with this biller.
`ejbFindAllPaymentTypes()`	Retrieves the payment type records associated with this biller.
`ejbFindAllAccountTypes()`	Retrieves the account type records associated with this biller.
`ejbCreateAccountType()`	Creates an account type record for this biller in the database.
`ejbRemoveAccountType()`	Removes an account type record for this biller from the database.
`ejbStoreAccountType()`	Updates an account type record for this biller in the database.
`ejbCreateFinancialAccount()`	Creates a financial account record for this biller in the database.
`ejbRemoveFinancialAccount()`	Removes a financial account record for this biller from the database.
`ejbStoreFinancialAccount()`	Updates a financial account record for this biller in the database.

Table 3-5 IBXBillerProfile Class Methods *(2 of 5)*

Methods	Description
`ejbCreatePaymentType()`	Creates a payment type record for this biller in the database.
`ejbRemovePaymentType()`	Removes a payment type record for this biller from the database.
`ejbStorePaymentType()`	Updates a payment type record for this biller in the database.
`ejbFindByOrganizationName`	Retrieves the organization name
`ejbFindAllBillers`	Retrieves all the billers
`ejbLoadAccountType`	Loads the biller's account types
`ejbLoadPaymentType`	Retrieves the biller's payment types
Getting and setting biller Profile information	
`getIG()`	Determines the IG user
`setIG()`	Specifies the IG user
`getIGURL()`	Determines the IG URL destination
`setIGURL()`	Sets the URL for the IG Server
`getType()`	Determines the kind of biller.
`setType()`	Specifies the kind of biller.
`getTabledefFilepath()`	Determines the path to the definition table
`setTabledefFilepath()`	Sets the path to the definition table
`getPrimaryContactPhone()`	Determines the primary phone #
`setPrimaryContactPhone()`	Sets the primary phone #
`getPrimaryContactEmail()`	Determines the primary email
`setPrimaryContactEmail()`	Setsthe primary email
`getSecondaryContactPhone()`	Determines the secondary phone
`setSecondaryContactPhone()`	Sets the secondary phone
`getSecondaryContactEmail()`	Determines the secondary email
`setSecondaryContactEmail()`	Sets the secondary email
`getEnrollOption()`	Determines the enrollment option
`setEnrollOption()`	Sets the enrollment option
`getBillPresentment()`	Determines which bills to present
`setBillPresentment()`	Sets what bills to present

Table 3-5 IBXBillerProfile Class Methods *(3 of 5)*

Methods	Description
`getAutopay()`	Determines the autopay option
`setAutopay()`	Sets the autopay option
`getOrgName()`	Determines the biller's organization name.
`setOrgName()`	Specifies the biller's organization name.
`getBusinessCategory()`	Determines the biller's business category.
`setBusinessCategory()`	Specifies the biller's business category.
`getDescription()`	Determines the biller's description.
`setDescription()`	Specifies the biller's description.
`getAddress()`	Determines the biller's address.
`setAddress()`	Specifies the biller's address.
`getPhone()`	Determines the biller's phone number.
`setPhone()`	Specifies the biller's phone number.
`getFaxNumber()`	Determines the biller's fax number.
`setFaxNumber()`	Specifies the biller's fax number.
`getEmail()`	Determines the biller's email address.
`setEmail()`	Specifies the biller's email address.
`getURL()`	Determines the biller's URL.
`setURL()`	Specifies the biller's URL.
`getLanguage()`	Determines the biller's language.
`setLanguage()`	Specifies the biller's language.
`getCurrency()`	Determines the biller's currency.
`setCurrency()`	Specifies the biller's currency.
`getCharSet()`	Determines the biller's character set.
`setCharSet()`	Specifies the biller's character set.
`getTimeZone()`	Determines the biller's time zone.
`setTimeZone()`	Specifies the biller's time zone.
`getBillerStatus()`	Determines the biller's status.
`setBillerStatus()`	Specifies the biller's status.
`getCreateDate()`	Determines the biller profile's creation date.

Table 3-5 IBXBillerProfile Class Methods *(4 of 5)*

Methods	Description
`setCreateDate()`	Sets the biller profile's creation date
`getModifyDate()`	Determines the biller profile's modification date.
`setModifyDate()`	Set the biller profile's modification date
`getPrimaryContact()`	Determines the biller's primary contact.
`setPrimaryContact()`	Specifies the biller's primary contact.
`getSecondaryContact()`	Determines the biller's secondary contacts.
`setSecondaryContact()`	Specifies the biller's secondary contacts.
`getProfileExtension()`	Determines the biller's profile extension.
`setProfileExtension()`	Specifies the biller's profile extension.
`getServiceExtension()`	Determines the biller's service extension.
`setServiceExtension()`	Specifies the biller's service extension.
`getCreatedBy()`	Determines the biller profile's creator.
`setCreatedBy()`	Specifies the biller profile's creator.
`getTaxID()`	Determines the biller's tax ID.
`setTaxID()`	Specifies the biller's tax ID.
`getExtension()`	Determines the biller's extension.
`setExtension()`	Specifies the biller's extension.
`getMerchantID()`	Determines the biller's merchant ID.
`setMerchantID()`	Specifies the biller's merchant ID.
`getPayeeID()`	Determines the payee's id
`setPayeeID()`	Sets the payee's id
Multiple Biller Extensions	
`getIndustryProfile()`	Determines the Industry profile of the billers
`setIndustryProfile()`	Sets the Industry profile of the billers
`getDatabase()`	Determines the biller database
`setDatabase()`	Sets the biller database
`getUserID()`	Determines the user id of the biller
`setUserID()`	Sets the user id of the biller
`getPassword()`	Determines the password of the biller

Table 3-5 IBXBillerProfile Class Methods *(5 of 5)*

Methods	Description
`setPassword()`	Sets the password of the biller
`getTemplatePath()`	Determines the path for a specific biller
`setTemplatePath()`	Sets the path for a specific biller
`getCycleDate()`	Determines the cycle date for billers
`setCycleDate()`	Sets the cycle date for billers
`getMonthHistory()`	Determines the history date
`setMonthHistory()`	Sets the history date
`getBillType()`	Determines the bill type for the billers
`setBillType()`	Sets the bill type for the billers
`getGroup()`	Determines the group of the billers
`SetGroup()`	Sets the billers group
`getMaxRows()`	Determines the maximum rows for the billers
`setMaxRows()`	Sets the maximum rows for the billers

ejbCreateAccountType()

Creates an account type record for this biller in the database.

Syntax

```
String ejbCreateAccountType (IBXAccountType accountType)
throws Exception;
```

Parameters

accountType — The IBXAccountType object to store in the database.

Discussion

Both the biller profile object's database context and the account type's database context must previously have been set by calling each object's `createMembershipContext()` method. The organization name must be set in the biller profile object and the type field must be set in the account type object. Returns a string that contains the object ID for the newly-created account type record.

ejbCreateFinancialAccount()

Creates a financial account record for this biller in the database.

Syntax

```
String ejbCreateFinancialAccount(IBXFinancialAccount account) throws Exception;
```

Parameters

account — The IBXFinancial Account object to store in the database.

Discussion

Both the biller profile object's database context and the account type's database context must previously have been set by calling each object's `createMembershipContext()` method. The organization name must be set in the biller profile object and the type field must be set in the financial type object.

Returns a string that contains the object ID for the newly-created financial account record.

ejbCreatePaymentType()

Creates a payment type record for this biller in the database.

Syntax

```
String ejbCreatePaymentType (IBXPaymentType paymentType) throws Exception;
```

Parameters

paymentType — The IBXPaymentType object to store in the database.

Discussion

Both the biller profile object's database context and the account type's database context must previously have been set by calling each object's `createMembershipContext()` method. The organization name must be set in the biller profile object and the type field must be set in the payment type object. Returns a string that contains the object ID for the newly-created account type record.

ejbFindAllCompanies()

Retrieves the companies belonging to this organization.

Syntax

```
public Vector ejbFindAllCompanies (BXFilter filter, BXOrder order) throws Exception;
```

Parameters

filter — The BXfilter object that specifies the Where clause in the SQL Select Statement.

order — A BXOrder object that specifies the Order By clause of an SQL Select Statement

Discussion

The `ejbFindAllCompanies()` method retrieves all records for the companies belonging to this organization that match the clause in the *filter* parameter, ordered by the clause specified in the *order* parameter. For each record, the `ejbFindAllCompanies()` method creates an object, sets the fields in the object from the record, and adds the object to the return vector.

If the *filter* parameter is null, all records will be retrieved. If the *order* parameter is null, the retrieved records will be sorted in ascending order by their primary keys. The customer profile object's database contact must previously been set by calling the customer profile object's `createMembershipContact()` method.

Returns a vector that contains the object BXCustomerProfile objects representing selected records.

ejbFindAllAccountTypes()

Retrieves the account type records associated with this biller.

Syntax

```
Vector ejbFindAllAccountTypes(BXFilter filter, BXOrder order) throws Exception;
```

Parameters

filter — The BXFilter object that specifies the Where clause of an SQL Select statement.

order — A BXOrder object that specifies the Order By clause of an SQL Select statement

Discussion

The `ejbFindAllAccountTypes()` method retrieves all records for the biller that match the clause in the *filter* parameter, ordered by the clause specified in the *order* parameter. For each record, the `ejbFindAllAccountTypes()` method creates an object, sets the fields in the object from the record, and adds the object to the return vector.

If the *filter* parameter is null, all records will be retrieved. If the *order* parameter is null, the retrieved records will be sorted in ascending order by their primary key. The biller profile object's database context must previously have been set by calling the biller profile object's `createMembershipContext()` method.

Returns a vector that contains the object BXAccountType objects representing the selected records.

ejbFindAllFinancialAccounts()

Retrieves the financial account records associated with this biller.

Syntax

```
Vector ejbFindAllFinancialAccounts(BXFilter filter, BXOrder order)
throws Exception;
```

Parameters *filter* — The BXFilter object that specifies the Where clause of an SQL Select statement.

order — A BXOrder object that specifies the Order By clause of an SQL Select statement

Discussion The `ejbFindAllFinancialAccounts()` method retrieves all records for the biller that match the clause in the *filter* parameter, ordered by the clause specified in the *order* parameter. For each record, the `ejbFindAllFinancialAccounts()` method creates an object, sets the fields in the object from the record, and adds the object to the return vector.

If the *filter* parameter is null, all records will be retrieved. If the *order* parameter is null, the retrieved records will be sorted in ascending order by their primary key.

The biller profile object's database context must previously have been set by calling the biller profile object's `createMembershipContext()` method. Returns a vector that contains the object BXAccountType objects representing the selected records.

ejbFindAllPaymentTypes()

Retrieves the payment type records associated with this biller.

Syntax

```
Vector ejbFindAllPaymentTypes(BXFilter filter, BXOrder order)
throws Exception;
```

Parameters *filter* — The BXFilter object that specifies the Where clause of an SQL Select statement.

order — A BXOrder object that specifies the Order By clause of an SQL Select statement.

Discussion The `ejbFindAllPaymentTypes()` method retrieves all records for the biller that match the clause in the *filter* parameter, ordered by the clause specified in the *order* parameter. For each record, the `ejbFindAllPaymentTypes()` method creates an object, sets the fields in the object from the record, and adds the object to the return vector.

If the *filter* parameter is null, all records will be retrieved. If the *order* parameter is null, the retrieved records will be sorted in ascending order by their primary key.

The biller profile object's database context must previously have been set by calling the biller profile object's `createMembershipContext()` method. Returns a vector that contains the object BXPaymentType objects representing the selected records.

ejbFindAllPersons()

Retrieves the customer profile records associated with this biller.

Syntax

```
Vector ejbFindAllPersons(BXFilter filter, BXOrder order)
throws Exception;
```

Parameters

filter — The BXFilter object that specifies the Where clause of an SQL Select statement.

order — A BXOrder object that specifies the Order By clause of an SQL Select statement.

Discussion

The `ejbFindAllPersons()` method retrieves all records for the biller that match the clause in the *filter* parameter, ordered by the clause specified in the *order* parameter. For each record, the `ejbFindAllPersons()` method creates an object, sets the fields in the object from the record, and adds the object to the return vector.

If the *filter* parameter is null, all records will be retrieved. If the ***order*** parameter is null, the retrieved records will be sorted in ascending order by their primary key.

The biller profile object's database context must previously have been set by calling the biller profile object's `createMembershipContext()` method.

Returns a vector that contains the object BXCustomerProfile objects representing the selected records.

ejbFindByOrganizationName()

Retrieves the biller profile record specified by the organization name.

Syntax

```
void ejbFindByOrganizationName (String name) throws Exception;
```

Parameters

name — A string that contains the name of the biller.

Discussion

The `name` parameter specifies the record to read. The fields of this object are set from the contents of the record. The biller profile object's database context must previously have been set by calling the biller profile object's `createMembershipContext()` method. This method is the same as the `ejbFindByPrimaryKey()` method.

Warning Previous contents of objects are replaced by contents of the selected record.

ejbRemoveAccountType()

Removes an account type record for this biller from the database.

Syntax	`void ejbRemoveAccountType( IBXAccountType` *accountType*`) throws Exception;`
Parameters	*accountType* — The IBXAccountType object to remove from the database
Discussion	The fields of the object itself are not changed. Both the biller profile object's database context and the account type object's database context must previously have been set by calling each object's `createMembershipContext()` method.

ejbRemoveFinancialAccount()

Removes a financial account record for this biller from the database.

Syntax	`void ejbRemoveFinancialAccount( IBXFinancialAccount` *Returns account* `) throws Exception;`
Parameters	*Returns account* — The IBXFinancialAccount object to remove from the database
Discussion	The fields of the object itself are not changed. Both the biller profile object's database context and the financial account object's database context must previously have been set by calling each object's `createMembershipContext()` method.

ejbRemovePaymentType()

Removes a payment type record for this biller from the database.

Syntax	`void ejbRemovePaymentType( IBXPaymentType` *paymentType*`) throws Exception;`
Parameters	*paymentType* — The IBXPaymentType object to remove from the database.
Discussion	The fields of the object itself are not changed. Both the biller profile object's database context and the financial account object's database context must previously have been set by calling each object's `createMembershipContext()` method.

ejbStoreAccountType()

Updates an account type record for this biller in the database.

Syntax `void ejbStoreAccountType( IBXAccountType` *Returns accountType*`) throws Exception;`

Parameters *Returns accountType* — The IBXFinancialAccountType to update from the database

Discussion Both the biller profile object's database context and the account type object's database context must previously have been set by calling each object's `createMembershipContext()` method.

Warning You must retrieve the record before you update it; otherwise, fields that you do not explicitly set will be written as null.

ejbStoreFinancialAccount()

Updates a financial account record for this biller in the database.

Syntax `void ejbStoreFinancialAccountType( IBXFinancialAccount` *Returns account*`) throws Exception;`

Parameters *Returns account* — The IBXFinancialAccount object to update from the database.

Discussion Both the biller profile object's database context and the account type object's database context must previously have been set by calling each object's `createMembershipContext()` method.

Warning You must retrieve the record before you update it; otherwise, fields that you do not explicitly set will be written as null.

ejbStorePaymentType()

Updates a payment type record for this biller in the database.

Syntax `void ejbStorePaymentType( IBXPaymentType` *paymentType*`) throws Exception;`

Parameters *paymentType* — The IBXPaymentType to update from the database

Discussion Both the biller profile object's database context and the account type object's database context must previously have been set by calling each object's `createMembershipContext()` method.

Warning You must retrieve the record before you update it; otherwise, fields that you do not explicitly set will be written as null.

ejbFindByOrganizationName

Retrieves billers by organization name.

Syntax

```
public void ejbFindByOrganizationName( String name) throws Exception;
```

Parameters *name* — A string that contains the name of the biller.

Discussion The *name* parameter specifies the record to read. The fields of this object are set from the contents of the record. The biller profile object's database context must previously have been set by calling the biller profile object's `createMembershipContext()` method. This method is the same as the `ejbFindByPrimaryKey()` method.

ejbFindAllBillers

Retrieves all the billers in the system.

Syntax

```
public Vector ejbFindAllBillers( BXFilter filter1, BXFilter filter2,
BXOrder order, String startIndex, int maxRows) throws Exception;
```

Parameters *filter1* — This is filter1 parameter

filter2 — This filter2 parameter

order — This is the order of biller discovery

startIndex — The beginning of the find

maxRows — The number of rows in the find.

Discussion

ejbLoadAccountType

Loads the Biller's account types.

Syntax	`public void ejbLoadAccountType( IBXAccountType` *Returns accountTyp*`e) throws Exception;`
Parameters	*Returns accountType* — The IBXAccountType to update in the database
Discussion	Both the biller profile object's database context and the account type object's database context must previously have been set by calling each object's `createMembershipContext()` method.

Warning	You must retrieve the record before you update it; otherwise, fields that you do not explicitly set will be written as null.

ejbLoadPaymentType

Retrieves the biller's payment types.

Syntax	`public void ejbLoadPaymentType( IBXPaymentType` *paymentType*`) throws Exception;`
Parameters	*paymentType* — The IBXPaymentType object to store in the database
Discussion	Both the biller profile object's database context and the account type object's database context must previously have been set by calling each object's `createMembershipContext()` method.

getAddress()

Determines the biller's address.

Syntax	`IBXAddress getAddress();`
Parameters	None.
Discussion	Returns an IBXAddress object that contains the address.

getBillerStatus()

Determines the biller's status.

Syntax	`String getBillerStatus();`

Parameters	None.
Discussion	Returns a string that contains the status.

getBusinessCategory()

Determines the biller's business category.

Syntax	`String getBusinessCategory();`
Parameters	None.
Discussion	Returns a string that contains the business category.

getCharSet()

Determines the biller's character set.

Syntax	`String getCharSet();`
Parameters	None.
Discussion	Returns a string that contains the character set.

getCreateDate()

Determines the biller profile's creation date.

Syntax	`Date getCreateDate();`
Parameters	None.
Discussion	Returns a string that contains the creation date.

getCreatedBy()

Determines the biller profile's creator.

Syntax	`String getCreatedBy();`
Parameters	None.
Discussion	Returns a string that contains the user name of the creator.

getCurrency()

Determines the biller's currency.

Syntax	`String getCurrency();`
Parameters	None.
Discussion	Returns a string that contains the currency.

getDescription()

Determines the biller's description.

Syntax	`String getDescription();`
Parameters	None.
Discussion	Returns a string that contains the description.

getEmail()

Determines the biller's email address.

Syntax	`String getEmail();`
Parameters	None.
Discussion	Returns a string that contains the email address.

getExtension()

Determines the biller's extension.

Syntax	`String getExtension();`
Parameters	None.
Discussion	Returns a string that contains the extension

getFaxNumber()

Determines the biller's fax number.

Syntax	`String getFaxNumber();`
Parameters	None.
Discussion	Returns a string that contains the fax number

getLanguage()

Determines the biller's language.

Syntax	`String getLanguage();`
Parameters	None.
Discussion	Returns a string that contains the language.

getMerchantID()

Determines the biller's merchant ID.

Syntax	`String getMerchantID();`
Parameters	None.
Discussion	Returns a string that contains the merchant ID.

getModifyDate()

Determines the biller profile's modification date.

Syntax	`String getModifyDate();`
Parameters	None.
Discussion	Returns a string that contains the modification date.

getOrgName()

Determines the biller's organization name.

Syntax	`String getOrgName();`
Parameters	None.
Discussion	Returns a string that contains the organization name.

getPhone()

Determines the biller's phone number.

Syntax	`String getPhone();`
Parameters	None.
Discussion	Returns a string that contains the biller's phone number.

getPrimaryContact()

Determines the biller's primary contact.

Syntax	`String getPrimaryContact();`
Parameters	None.
Discussion	Returns a string that contains the primary contact.

getProfileExtension()

Determines the biller's profile extension.

Syntax	`String getProfileExtension();`
Parameters	None.
Discussion	Returns a string that contains the profile extension.

getSecondaryContact()

Determines the biller's secondary contacts.

Syntax	`Vector getSecondaryContact();`
Parameters	None.
Discussion	Returns a vector that contains the secondary contact.

getServiceExtension()

Determines the biller's service extension.

Syntax	`String getServiceExtension();`
Parameters	None.
Discussion	Returns a string that contains the service extension.

getTaxID()

Determines the biller's tax ID.

Syntax	`String getTaxID();`
Parameters	None.
Discussion	Returns a string that contains the tax ID.

getTimeZone()

Determines the biller's time zone.

Syntax	`String getTimeZone();`
Discussion	Returns a string that contains the time zone.

getType()

Determines the kind of biller.

Syntax	`String getType();`

Parameters	None.
Discussion	Returns a string that contains the biller type.

getURL()

Determines the biller's URL.

Syntax	`String getURL();`
Parameters	None.
Discussion	Returns a string that contains the biller's URL.

setAddress()

Specifies the biller's address .

Syntax	`void setAddress( IBXAddress` *value*`);`
Parameters	*value* — An IBXAddress object that specifies the address
Discussion	This method only changes the value of the address field in the object; it does not modify the value in the database.

setBillerStatus()

Specifies the biller's status.

Syntax	`void setBillerStatus( String` *value*`);`
Parameters	*value* — An object that specifies the status
Discussion	This method only changes the value of the status field in the object; it does not modify the value in the database.

setBusinessCategory()

Specifies the biller's business category.

Syntax	`void setBusinesscategory (String` *value*`);`
Parameters	*value* — A string that specifies the category

Discussion	This method only changes the value of the address field in the object; it does not modify the value in the database.

setCharSet()

Specifies the biller's character set.

Syntax	`void setCharSet (String` *value*`);`
Parameters	*value* — A string that specifies the character set
Discussion	This method only changes the value of the character set field in the object; it does not modify the value in the database.

setCreatedBy()

Specifies the biller profile's creator.

Syntax	`void setCreatedBy (String` *value*`);`
Parameters	*value* — A object that specifies the creator
Discussion	This method only changes the value of the address field in the object; it does not modify the value in the database.

setCurrency()

Specifies the biller's currency.

Syntax	`void setCurrency( String value);`
Parameters	*value* — A string that specifies the currencye
Discussion	This method only changes the value of the currency field in the object; it does not modify the value in the database.

setDescription()

Specifies the biller's description.

Syntax	`void setDescription (String` *value*`);`
Parameters	value — A string that specifies the description

Discussion	This method only changes the value of the description field in the object; it does not modify the value in the database.

setEmail()

Specifies the biller's email address.

Syntax	`void setEmail (String` *value*`);`
Parameters	*value* — A string that specifies the email address
Discussion	This method only changes the value of the email field in the object; it does not modify the value in the database.

setExtension()

Specifies the biller's extension.

Syntax	`void setExtension (String` *value*`);`
Parameters	value — A string that specifies the email address
Discussion	This method only changes the value of the extension field in the object; it does not modify the value in the database.

setFaxNumber()

Specifies the biller's fax number.

Syntax	`void setFaxNumber (String` *value*`);`
Parameters	*value* — A string that specifies the fax number
Discussion	This method only changes the value of the fax number field in the object; it does not modify the value in the database.

setLanguage()

Specifies the biller's language.

Syntax	`void setLanguage (String` *value*`);`
Parameters	*value* — A string that specifies the language

Discussion	This method only changes the value of the language field in the object; it does not modify the value in the database.

setMerchantID()

Specifies the biller's merchant ID.

Syntax	`void setMerchantID (String` *value*`);`
Parameters	*value* — A string that specifies the merchant ID
Discussion	This method only changes the value of the ID field in the object; it does not modify the value in the database.

setOrgName()

Specifies the biller's organization name.

Syntax	`void setOrgName (String` *value*`);`
Parameters	*value* — A string that specifies the organization name
Discussion	This method only changes the value of the organization name field in the object; it does not modify the value in the database.

setPhone()

Specifies the biller's phone number.

Syntax	`void setPhone(String` *value*`);`
Parameters	*value* — A string that specifies the phone number.
Discussion	This method only changes the value of the phone number field in the object; it does not modify the value in the database.

setPrimaryContact()

Specifies the biller's primary contact.

Syntax	`void setPrimaryContact (String` *value*`);`
Parameters	*value* — A string that specifies the contact

Discussion	This method only changes the value of the contact field in the object; it does not modify the value in the database.

setProfileExtension()

Specifies the biller's profile extension.

Syntax	`void setProfileExtension (String` *value*`);`
Parameters	*value* — A string that specifies the extension
Discussion	This method only changes the value of the extension field in the object; it does not modify the value in the database.

setSecondaryContact()

Specifies the biller's secondary contacts.

Syntax	`void setSecondaryContact (Vector` *value*`);`
Parameters	*value* — A string that specifies the secondary contact
Discussion	This method only changes the value of the contact field in the object; it does not modify the value in the database.

setServiceExtension()

Specifies the biller's service extension.

Syntax	`void setServiceExtension (String` *value*`);`
Parameters	*value* — A string that specifies the service extension
Discussion	This method only changes the value of the extension field in the object; it does not modify the value in the database.

setTaxID()

Specifies the biller's tax ID.

Syntax	`void setTaxID (String` *value*`);`
Parameters	*value* — A string that specifies the tax ID.

Discussion	This method only changes the value of the tax ID field in the object; it does not modify the value in the database.

setTimeZone()

Determines the biller's time zone.

Syntax	`void setTimeZone (String` *value*`);`
Parameters	*value* — A string that specifies the time zone
Discussion	This method only changes the value of the time zone field in the object; it does not modify the value in the database.

setType()

Specifies the kind of biller.

Syntax	`void setType (String` *value*`);`
Parameters	*value* — A string that specifies the type
Discussion	This method only changes the value of the type field in the object; it does not modify the value in the database.

setURL()

Specifies the biller's URL.

Syntax	`void setURL (String` *value*`);`
Parameters	*value* — A string that specifies the URL
Discussion	This method only changes the value of the URL field in the object; it does not modify the value in the database.

BXCustomerProfile Class Reference

The BXCustomerProfile class is an implementation of the IBXCustomerProfile interface, which defines customer profile objects.

Whenever possible, use the interface class IBXCustomerProfile (see "IBXCustomerProfile Class Reference") instead of BXCustomerProfile.

IBXCustomerProfile Class Reference

The IBXCustomerProfile interface defines customer profile objects that represent customer profile records. You call the `BXMembershipMgr.createBillerProfile()` method to create a biller profile object.

Before you can call the `IBXEntity.ejbCreate()` method to create a new record Returns associated with the object, you must call the setLoginName(), `setPassword()`, `setLastName()`, `setFirstName()`, and `setOrgName()` methods to set the login name, password, last name, first name, and organization name fields, respectively.

Interface Definition

Table 3-6 describes the IBXCustomerProfile interface definition.

Table 3-6 IBXCustomerProfile Interface Definition

Package	`com.iplanet.ecomm.billxb2b.ejb.membership;`
Syntax	`interface IBXCustomerProfile extends IBXMembershipEntity {...}`

Methods

This section describes the IBXCustomerProfile methods. Table 3-7 lists these methods

Table 3-7 IBXCustomerProfile Interface Methods *(1 of 4)*

Method	Description
Manipulating customer-related records	
`ejbFindByLoginName()`	Retrieves the record with the specified login name.
`ejbFindAllFinancialAccounts()`	Retrieves financial account records associated with this customer.
`ejbFindAllByIDList()`	Retrieves customer profile records with the specified IDs.
`ejbCreateFinancialAccount()`	Creates a financial account record for this customer.

Table 3-7 IBXCustomerProfile Interface Methods *(2 of 4)*

Method	Description
`ejbRemoveFinancialAccount()`	Removes a financial account record associated with this customer.
`ejbLoadFinancialAccount()`	Loads a financial acount record
`ejbStoreFinancialAccount()`	Updates a financial account record associated with this customer.
Returns authenticating and protecting customer records	
`Returns authenticate()`	Verifies a customer's login name and password.
`updatePassword()`	Changes the customer's password.
Getting and setting customer profile information	
`getBillerID()`	Determines the Biller ID.
`getAccount()`	Determines the customer's account number.
`setAccount()`	Specifies the customer's account number.
`getPrefix()`	Determines the customer's name prefix.
`setPrefix()`	Specifies the customer's name prefix.
`getSuffix()`	Determines the customer's name suffix.
`setSuffix()`	Specifies the customer's name suffix.
`getLastName()`	Determines the customer's surname.
`setLastName()`	Specifies the customer's surname.
`getFirstName()`	Determines the customer's given name.
`setFirstName()`	Specifies the customer's given name.
`getMiddleName()`	Determines the customer's middle name.
`setMiddleName()`	Specifies the customer's middle name.
`getNickName()`	Determines the customer's nick name.
`setNickName()`	Specifies the customer's nick name.
`getDescription()`	Determines the customer's description.
`setDescription()`	Specifies the customer's description.
`getAddress()`	Determines the customer's address.
`setAddress()`	Specifies the customer's address.
`getDayPhone()`	Determines the customer's daytime phone number.

Table 3-7 IBXCustomerProfile Interface Methods *(3 of 4)*

Method	Description
`setDayPhone()`	Specifies the customer's daytime phone number.
`getNightPhone()`	Determines the customer's evening phone number.
`setNightPhone()`	Specifies the customer's evening phone number.
`getFaxNumber()`	Determines the customer's fax number.
`setFaxNumber()`	Specifies the customer's fax number.
`getEmail()`	Determines the customer's email address.
`setEmail()`	Specifies the customer's email address.
`getSecretWord()`	Determines the customer's secret word.
`setSecretWord()`	Specifies the customer's secret word.
`getLoginName()`	Determines the customer's login name.
`setLoginName()`	Specifies the customer's login name.
`getPassword()`	Determines the customer's password.
`setPassword()`	Specifies the customer's password.
`getPasswordHint()`	Determines the customer's password hint.
`setPasswordHint()`	Specifies the customer's password hint.
`getPasswordLastChanged()`	Determines the date the customer's password was last changed.
`setPasswordLastChanged()`	Specifies the date the customer's password was last changed.
`getLanguage()`	Determines the customer's language.
`setLanguage()`	Specifies the customer's language.
`getCurrency()`	Determines the customer's currency.
`setCurrency()`	Specifies the customer's currency.
`getCharSet()`	Determines the customer's character set.
`setCharSet()`	Specifies the customer's character set.
`getTimeZone()`	Determines the customer's time zone.
`setTimeZone()`	Specifies the customer's time zone.
`getCustomerStatus()`	Determines the customer's status.
`setCustomerStatus()`	Specifies the customer's status.
`getCreateDate()`	Determines the customer profile's creation date.
`getModifyDate()`	Determines the customer profile's modification date.

Table 3-7 IBXCustomerProfile Interface Methods *(4 of 4)*

Method	Description
`getCreatedBy()`	Determines the customer profile's creator.
`setCreatedBy()`	Specifies the customer profile's creator.
`getPrimaryContact()`	Determines the customer's primary contact.
`setPrimaryContact()`	Specifies the customer's primary contact.
`getSecondaryContact()`	Determines the customer's secondary contacts.
`setSecondaryContact()`	Specifies the customer's secondary contacts.
`getProfileExtension()`	Determines the customer's profile extension.
`setProfileExtension()`	Specifies the customer's profile extension.
`getServiceExtension()`	Determines the customer's service extension.
`setServiceExtension()`	Specifies the customer's service extension.
`getOrgName()`	Determines the customer's organization name.
`setOrgName()`	Specifies the customer's organization name.
`getExtension()`	Determines the customer's extension.
`setExtension()`	Specifies the customer's extension.
`getGroup()`	Determines the customer's groups.
`setGroup()`	Specifies the customer's groups.
`getAccountType()`	Determines the customer's account types.
`setAccountType()`	Specifies the customer's account types.
`getDefaultPmtAcct()`	Determines the customer default payment option.
`setDefaultPmtAcct()`	Sets the customer default payment option.
`getFileID()`	Determines the file ID.
`setFileID()`	Specifies the file ID.
Generic Methods	
`setAttribute()`	Specifies the attribute.
`callCreateAdmin()`	Creates an admin for the customer profile.
`callStoreAdmin()`	Stores an admin for the customer profile.

Returns authenticate()

Verifies a customer's login name and password.

Syntax	`String authenticate (String` *`login`*`, String` *`password`*`) throws Exception;`
Parameters	*login* — A string that contains the login name. *password* — A string that contains the password
Discussion	The `authenticate()` method throws an exception if the login name does not match the password.

callCreateAdmin()

Creates an Admin for a customer profile.

Syntax	`void callCreateAdmin ( BXStatus` *`status`*`) throws Exception;`
Parameters	*status* — A BXStatus object
Discussion	The `callCreateAdmin()` method throws an exception if an Admin is not created.

callStoreAdmin()

Stores an Admin for a customer profile.

Syntax	`void callStoreAdmin () throws Exception;`
Parameters	None.
Discussion	The `callStoreAdmin()` method throws an exception if an Admin is not stored.

ejbCreateFinancialAccount()

Creates a financial account record for this customer.

Syntax	`String ejbCreateFinancialAccount(IBXFinancialAccount` *`Returns accoun`*`t) throws Exception;`
Parameters	*Returns account* — An IBXFinancialAccount object to store in the database

Discussion	The customer profile object's database context must previously have been set by calling the customer profile object's `createMembershipContext()` method. The organization name must be set in the customer profile object and the type, payment handle, and pan handle fields must be set in the financial account object. Returns a string that contains the object ID for the newly-created financial account record.

ejbFindAllByIDList()

Retrieves customer profile records with the specified IDs.

Syntax	`Vector ejbFindAllByIDList(Vector` *idList*`)throws Exception;`
Parameters	*idList* — A vector object that specifies the ID
Discussion	The `ejbFindAllByIDList()` method retrieves all records for the customers whose uid fields match an element of the idList parameter, in ascending order. For each record, the `ejbFindAllByIDList()` method creates an object, sets the fields in the object from the record, Returns and adds the object to the return vector. The customer profile object's database context must previously have been set by calling the customer profile object's `createMembershipContext()` method. Returns a vector that contains BXCustomerProfile objects representing the selected records.

ejbFindAllFinancialAccounts()

Retrieves customer profile records with the specified IDs.

Syntax	`Vector ejbFindAllFinancialAccounts(BXFilter` *filter*`, BXOrder` *order*`) throws Exception;`
Parameters	*filter* — A BXFilter object that specifies the Where clause of an SQL Select statement. *order* — A BXOrder object that species the Order By clause of an SQL Select statement.

Discussion

The `ejbFindAllFinancialAccounts()` method retrieves all records for the customer that match the clause in the *filter* parameter, ordered by the clause specified in the *order* parameter. For each record, the `ejbFindAllFinancialAccounts()` method creates an object, sets the fields in the object from the record, and adds the object to the return vector.

If the *filter* parameter is null, all records will be retrieved. If the *order* parameter is null, the retrieved records will be sorted in Returns ascending order by their primary key.

Both the customer profile object's database context and the financial Returns account object's database context must previously have been set by calling each object's `createMembershipContext()` method. Returns a vector that contains `BXFinancialAccount` objects representing the selected records.

ejbFindByLoginName()

Retrieves the record with the specified login name.

Syntax void ejbFindByLoginName (String *loginName*) throws Exception;

Parameters *loginName* — A string that contains the login name

Discussion The *loginName* parameter specifies the record to read. The fields of this object are set from the contents of the record. The customer profile object's database context must previously have been set by calling the customer profile object's `createMembershipContext()` method. This method is the same as the `ejbFindByPrimaryKey()` method.

Warning The previous contents of this object are replaced by the contents of the selected record.

ejbRemoveFinancialAccount()

Removes a financial account record associated with this customer.

Syntax `void ejbRemoveFinancialAccount(IBXFinancialAccount` *Returns_account*`) throws Exception;`

Parameters *Returns_account* — The IBXFinancialAccount object to remove from the database

Discussion	The fields of the object itself are not changed. Both the customer profile object's database context and the financial account object's database context must previously have been set by calling each object's `createMembershipContext()` method.

ejbLoadFinancialAccount()

Loads a financial account.

Syntax	`void ejbLoadFinancialAccount ( IBXFinancialAccount` *Returns_account*`) throws Exception;`
Parameters	*Returns_account* — The IBXFinancialAccount object to load from the database
Discussion	The fields of the object itself are not changed. Both the customer profile object's database context and the financial account object's database context must previously have been set by calling each object's `createMembershipContext()` method.

ejbStoreFinancialAccount()

Updates a financial account record associated with this customer.

Syntax	`void ejbStoreFinancialAccount(IBXFinancialAccount` *Returns_account* `) throws Exception;`
Parameters	*Returns_account* — The IBXFinancialAccount object to update in the database
Discussion	Both the customer profile object's database context and the financial account object's database context must previously have been set by calling each object's `createMembershipContext()` method.

Important	You must retrieve the record before you update, otherwise fields that you do not explicitly set will be written as null.

getAccount()

Determines the customer's account number.

Syntax	`String getAccount();`
Parameters	None.

Discussion	Returns a string that contains the account number.

getAccountType()

Determines the customer's account types.

Syntax	`String getAccountType();`
Parameters	None.
Discussion	Returns a vector that contains the account type.

getAddress()

Determines the customer's address.

Syntax	`IBXAddress getAddress();`
Parameters	None.
Discussion	Returns a IBXAddress object that contains the address.

getCharSet()

Determines the customer's character set.

Syntax	`String getCharSet();`
Parameters	None.
Discussion	Returns a string that contains the character set.

getCreateDate()

Determines the customer profile's creation date.

Syntax	`Date getCreateDate();`
Parameters	None.
Discussion	Returns a Date object that contains the creation date.

getCreatedBy()

Determines the customer profile's creator.

Syntax	`String getCreatedBy();`
Parameters	None.
Discussion	Returns a string that contains the account's creator.

getCurrency()

Determines the customer's currency.

Syntax	`String getCurrency();`
Parameters	None.
Discussion	Returns a string that contains the currency

getCustomerStatus()

Determines the customer's status.

Syntax	`String getCustomerStatus();`
Parameters	None.
Discussion	Returns a string that contains the customer's status.

getDayPhone()

Determines the customer's daytime phone number.

Syntax	`String getDayPhone();`
Parameters	None.
Discussion	Returns a string that contains the phone number.

getDefaultPmtAcct()

Determines the default payment account.

Syntax	`String getDefaultPmtAcct();`
Parameters	None.
Discussion	Returns a string that contains the default payment account.

getDescription()

Determines the customer's description.

Syntax	`String getDescription();`
Parameters	None.
Discussion	Returns a string that contains the description..

getEmail()

Determines the customer's email address.

Syntax	`String getEmail();`
Parameters	None.
Discussion	Returns a string that contains the email address..

getExtension()

Determines the customer's extension.

Syntax	`String getExtension();`
Parameters	None.
Discussion	Returns a string that contains the extension.

getFaxNumber()

Determines the customer's fax number.

Syntax	`String getFaxNumber();`
Parameters	None.
Discussion	Returns a string that contains the number.

getFirstName()

Determines the customer's given name.

Syntax	`String getFirstName();`
Parameters	None.
Discussion	Returns a string that contains the name.

getGroup()

Determines the customer's groups.

Syntax	`public Vector getGroup();`
Parameters	None.
Discussion	Returns a string that contains the customer's group(s)..

getLanguage()

Determines the customer's language.

Syntax	`String getLanguage();`
Parameters	None.
Discussion	Returns a string that contains the language.

getLastName()

Determines the customer's surname.

Syntax	`String getLastName();`
Parameters	None.
Discussion	Returns a string that contains the customer's last name.

getLoginName()

Determines the customer's login name.

Syntax	`String getLoginName();`
Parameters	None.
Discussion	Returns a string that contains the login name.

getMiddleName()

Determines the customer's middle name.

Syntax	`String getMiddleName();`
Parameters	None.
Discussion	Returns a string that contains the name.

getModifyDate()

Determines the modification date.

Syntax	`String getModifyDate();`
Parameters	None.
Discussion	Returns a string that contains the modification date.

getNickName()

Determines the customer's nick name.

Syntax	`String getNickName();`
Parameters	None.
Discussion	Returns a string that contains the name.

getNightPhone()

Determines the customer's evening phone number.

Syntax	`String getNightPhone();`
Parameters	None.
Discussion	Returns a string that contains the number.

getOrgName()

Determines the customer's organization name.

Syntax	`String getOrgName();`
Parameters	None.
Discussion	Returns a string that contains the name.

getPassword()

Specifies the customer's password.

Syntax	`String getPassword();`
Parameters	None.
Discussion	Returns a string that contains the password.

getPasswordHint()

Determines the customer's password hint.

Syntax	`String getPasswordHint();`
Parameters	None.
Discussion	Returns a string that contains the word.

getPasswordLastChanged()

Determines the date the customer's password was last changed.

Syntax	`Date getPasswordLastChanged();`
Parameters	None.
Discussion	Returns a Date object that contains the date.

getPrefix()

Determines the customer's name prefix.

Syntax	`String getPrefix();`
Parameter	None.
Discussion	Returns a string that contains the prefix.

getPrimaryContact()

Determines the customer's primary contact.

Syntax	`String getPrimaryContact();`
Parameters	None.
Discussion	Returns a string that contains the contact.

getProfileExtension()

Determines the customer's profile extension.

Syntax	`String getProfileExtension();`
Parameters	None.
Discussion	Returns a string that contains the profile extension.

getSecondaryContact()

Determines the customer's secondary contacts.

Syntax	`Vector getSecondaryContact();`
Parameters	None.
Discussion	Returns a vector that contains the contact(s).

getSecretWord()

Determines the customer's secret word.

Syntax	`String getSecretWord();`
Parameters	None.
Discussion	Returns a string that contains the word.

getServiceExtension()

Determines the customer's service extension.

Syntax	`String getServiceExtension();`
Parameters	None.
Discussion	Returns a string that contains the number.

getSuffix()

Determines the customer's name suffix.

Syntax	`String getSuffix();`
Parameters	None.
Discussion	Returns a string that contains the suffix.

getTimeZone()

Determines the customer's time zone.

Syntax	`String getTimeZone();`
Parameters	None.
Discussion	Returns a string that contains the time zone.

setAccount()

Specifies the customer's account number.

Syntax	`void setAccount (String` *value*`);`
Parameters	*value* — A string that specifies the account
Discussion	This method only changes the value of the account number in the object; it does not modify the value in the database.

setAccountType()

Specifies the customer's account types.

Syntax	`void setAccountType (Vector` *value*`);`
Parameters	*value* — A vector that specifies the account type
Discussion	This method only changes the value of the account type in the object; it does not modify the value in the database.

setIBXAddress()

Specifies the customer's address.

Syntax	`void setIBXAddress (IBXAddress` *value*`);`
Parameters	*value* — An IBXAdress object that specifies the address.
Discussion	This method only changes the value of the address field in the object; it does not modify the value in the database.

setAttribute()

Specifies a generic attribute.

Syntax	`void setAttribute (String` *name*`, String` *value*`);`
Parameters	*name* — A string that specifies the attribute name. *value* — A string that specifies the address.
Discussion	This method only changes the value of the attribute field in the object; it does not modify the value in the database.

setCharSet()

Specifies the customer's character set.

Syntax	`void setCharSet (String` *value*`);`
Parameters	*value* — A string that specifies the character set.
Discussion	This method only changes the value of the character set field in the object; it does not modify the value in the database.

setCreatedBy()

Determines the customer profile's creator.

Syntax	`void setCreatedBy (String` *value*`);`
Parameters	*value* — A string that specifies the creator.
Discussion	This method only changes the value of the in the creator field in the object; it does not modify the value in the database.

setCurrency()

Specifies the customer's currency.

Syntax	`void setCurrency (String value);`
Parameters	*value* — A string that specifies the currency
Discussion	This method only changes the value of the currency in the object; it does not modify the value in the database.

setCustomerStatus()

Specifies the customer's status.

Syntax	`void setCustomerStatus (String value);`
Parameters	*value* — A string that specifies the status
Discussion	This method only changes the value of the status in the object; it does not modify the value in the database.

setDayPhone()

Specifies the customer's daytime phone number.

Syntax	`void setDayPhone (String value);`
Parameters	*value* — A string that specifies the daytime phone number
Discussion	This method only changes the value of the phone number in the object; it does not modify the value in the database.

setDefaultPmtAcct()

Specifies the default payment account.

Syntax	`void setDefaultPmtAcct (String value);`
Parameters	*value* — A string that specifies the default payment account
Discussion	This method only changes the value of the description field in the object; it does not modify the value in the database.

setDescription()

Specifies the customer's description.

Syntax	`void setDescription (String` *value*`);`
Parameters	*value* — A string that specifies the description
Discussion	This method only changes the value of the description in the object; it does not modify the value in the database.

setEmail()

Specifies the customer's email address.

Syntax	`void setEmail (String` *value*`);`
Parameters	*value* — A string that specifies the email address
Discussion	This method only changes the value of the email address in the object; it does not modify the value in the database.

setExtension()

Specifies the customer's extension.

Syntax	`void setExtension (String` *value*`);`
Parameters	*value* — A string that specifies the extension
Discussion	This method only changes the value of the extension in the object; it does not modify the value in the database.

setFaxNumber()

Specifies the customer's fax number.

Syntax	`void setFaxNumber (String` *value*`);`
Parameters	*value* — A string that specifies the fax number
Discussion	This method only changes the value of the number in the object; it does not modify the value in the database.

setFileID()

Specifies the file ID.

Syntax	`void setFileID (String` *value*`);`
Parameters	*value* — A string that specifies the file ID
Discussion	This method only changes the value of the file ID in the object; it does not modify the value in the database.

setFirstName()

Specifies the customer's given name.

Syntax	`void setFirstName (String` *value*`);`
Parameters	*value* — A string that specifies the name
Discussion	This method only changes the value of the name in the object; it does not modify the value in the database.

setGroup()

Specifies the customer's groups.

Syntax	`void setGroup (Vector` *value*`);`
Parameters	*value* — A string that specifies the group(s)
Discussion	This method only changes the value of the group in the object; it does not modify the value in the database.

setLanguage()

Specifies the customer's language.

Syntax	`void setLanguage (String` *value*`);`
Parameters	*value* — A string that specifies the language
Discussion	This method only changes the value of the language in the object; it does not modify the value in the database.

setLastName()

Specifies the customer's surname.

Syntax	`void setLastName (String` *value*`);`
Parameters	*value* — A string that specifies the name
Discussion	This method only changes the value of the name field the object; it does not modify the value in the database.

setLoginName()

Specifies the customer's login name.

Syntax	`void setLoginName (String` *value*`);`
Parameters	*value* — A string that specifies the login name
Discussion	This method only changes the value of the login name field in the object; it does not modify the value in the database.

setMiddleName()

Specifies the customer's middle name.

Syntax	`void setMiddleName (String` *value*`);`
Parameters	*value* — A string that specifies the name
Discussion	This method only changes the value of the middle name field in the object; it does not modify the value in the database.

setNickName()

Specifies the customer's nick name.

Syntax	`void setNickName (String` *value*`);`
Parameters	*value* — A string that specifies the name
Discussion	This method only changes the value of the nickname field in the object; it does not modify the value in the database.

setNightPhone()

Specifies the customer's evening phone number.

Syntax	`void setNightPhone (String value);`
Parameters	*value* — A string that specifies the number
Discussion	This method only changes the value of the number field in the object; it does not modify the value in the database.

setOrgName()

Specifies the customer's organization name.

Syntax	`void setOrgName (String value);`
Parameters	*value*— A string that specifies the name
Discussion	This method only changes the value of the name field in the object; it does not modify the value in the database.

setPassword()

Specifies the customer's password.

Syntax	`public void setPassword (String value);`
Parameters	*value* — A string that specifies the password
Discussion	This method only changes the value of the password field in the object; it does not modify the value in the database.

setPasswordHint()

Specifies the customer's password hint.

Syntax	`void setPasswordHint (String value);`
Parameters	*value* — A string that specifies the password hint the customer provided.
Discussion	This method only changes the value of the word field in the object; it does not modify the value in the database.

setPasswordLastChanged()

Specifies the date the customer's password was last changed.

Syntax	`public void setPasswordLastChanged (Date` *value*`);`
Parameters	*value* — A string that specifies the date
Discussion	This method only changes the value of the date field in the object; it does not modify the value in the database.

setPrefix()

Specifies the customer's name prefix.

Syntax	`void setPrefix (String` *value*`);`
Parameters	*value* — A string that specifies the prefix.
Discussion	This method only changes the value of the prefix field in the object; it does not modify the value in the database.

setPrimaryContact()

Specifies the customer's primary contact.

Syntax	`void setPrimaryContact (String` *value*`);`
Parameters	*value* — A string that specifies the customer's primary contact.
Discussion	This method only changes the value of the contact field in the object; it does not modify the value in the database.

setProfileExtension()

Specifies the customer's profile extension.

Syntax	`public void setProfileExtension (String` *value*`)`
Parameters	*value* — A string that specifies the profile extension
Discussion	This method only changes the value of the profile extension field in the object; it does not modify the value in the database.

setSecondaryContact()

Specifies the customer's secondary contacts.

Syntax	`void setSecondaryContact (String` *value*`);`
Parameters	*value* — A string that specifies the contact
Discussion	This method only changes the value of the contact field in the object; it does not modify the value in the database.

setSecretWord()

Specifies the customer's secret word.

Syntax	`void setSecretWord (String` *value*`);`
Parameters	*value* — A string that specifies the word
Discussion	This method only changes the value of the word field in the object; it does not modify the value in the database.

setServiceExtension()

Specifies the customer's service extension.

Syntax	`void setServiceExtension (String` *value*`);`
Parameters	*value* — A string that specifies the extension
Discussion	This method only changes the value of the extension field in the object; it does not modify the value in the database.

setSuffix()

Specifies the customer's name suffix.

Syntax	`void setSuffix (String` *value*`);`
Parameters	*value* — A string that specifies the suffix
Discussion	This method only changes the value of the suffix field in the object; it does not modify the value in the database.

setTimeZone()

Specifies the customer's time zone.

Syntax	`void setTimeZone (String` *value*`);`
Parameters	*value* — A string that specifies the time zone
Discussion	This method only changes the value of the zone field in the object; it does not modify the value in the database.

updatePassword()

Changes the customer's password.

Syntax	`void updatePassword (String` *loginName*`, String` *password*`) throws Exception;`
Parameters	*loginName* — A string that specifies the login name. *value* — A string that specifies the word.
Discussion	The `updatePassword()` method changes the password associated with the specified login name. The organization field must be set in the object and the login name for that organization must exist on the server. This method sets the values of the password, password last changed, and modification date fields in the object and updates the values of these fields in the database.

IBXFinancialAccount Class Reference

The BXFinancialAccount class is an implementation of the IBXFinancialAccount interface, which defines customer profile objects that represent financial account records.

Whenever possible, use the interface class IBXFinancialAccount (see "IBXFinancialAccount Class Reference") instead of BXFinancialAccount.

IBXFinancialAccount Class Reference

The IBXFinancialAccount interface defines customer profile objects that represent financial account records. You call the `BXMembershipMgr.createFinancialAccount()` method to create a financial Returns account object.

Before you can call the `IBXEntity.ejbCreate()` method to create a new record Returns associated with the object, you must call the `setPayment()` method to set the payment handle field.

Interface Definition

Table 3-8 describes the IBXFinancialAccount interface definition.

Table 3-8 IBXFinancialAccount Interface Definition

Package	`com.iplanet.ecomm.billxb2b.ejb.membership;`
Syntax	`public class IBXFinancialAccount extends IBXMembershipEntity {...}`

Methods

This section describes the IBXFinancialAccount interface methods.

Table 3-9 IBXFinancialAccount Interface Methods

Method	Description
Getting and setting financial	**Returns account information**
`getPaymentHandle()`	Determines the financial account's payment type.
`setPaymentHandle()`	Specifies the financial account's payment type.
`getDescription()`	Determines the financial account's description.
`setDescription()`	Specifies the financial account's description.
`getType()`	Determines the financial account's payment type.
`setType()`	Specifies the financial account's payment type.
`getAlias()`	Determines the user-defined name associated with this financial account.
`setAlias()`	Specifies the user-defined name associated with this financial account.
`getPaymentHandle()`	Determines the financial account's payment type.
`getExtension()`	Determines the extension associated with this financial account
`setExtension()`	Specifies the extension associated with this financial account
`getRoutingNumber()`	Determines the routing number associated with this financial institution.
`setRoutingNumber()`	Specifies the routing number associated with this financial institution.
`getAccountNumber()`	Determines the account number associated with this financial institution
`setAccountNumber()`	Specifies the number associated with this financial institution

getAccountNumber()

Determines the account number.

Syntax	`String getAccountNumber();`
Parameters	None.
Discussion	Returns a string that contains the account number.

getAlias()

Determines the user-defined name associated with this financial account.

Syntax	`String getAlias();`
Parameters	None.
Discussion	Returns a string that contains the alias.

getDescription()

Determines the financial account's description.

Syntax	`String getDescription();`
Parameters	None.
Discussion	Returns a string that contains the description.

getExtension()

Determines the extension associated with this financial account.

Syntax	`String getExtension();`
Parameters	None.
Discussion	Returns a string that contains the description.

getPaymentHandle()

Determines the financial account's payment type.

Syntax	`String getPaymentHandle();`
Parameters	None.
Discussion	Returns a string that contains the handle.

getRoutingNumber()

Determines the routing number.

Syntax	`String getRoutingNumber();`
Parameters	None.
Discussion	Returns a string that contains the routing number.

getType()

Determines the financial account's payment type.

Syntax	`String getType();`
Parameters	None.
Discussion	Returns a string that contacts the type of payment that will be made.

setAlias()

Specifies the user-defined name associated with this financial account.

Syntax	`void setAlias (String` *value*`);`
Parameters	*value* — A string that specifies the alias
Discussion	This method only changes the value of the alias field in the object; it does not modify the value in the database.

setDescription()

Specifies the financial account's description.

Syntax	`void setDescription (String` *value*`);`
Parameters	*value* — A string that specifies thedescription
Discussion	This method only changes the value of the description field in the object; it does not modify the value in the database.

setPaymentHandle()

Specifies the financial account's payment type.

Syntax	`void setPaymentHandle (String` *value*`);`
Parameters	*value* — A string that specifies the handle
Discussion	This method only changes the value of the handle field in the object; it does not modify the value in the database.

setRoutingNumber()

Specifies the financial account's routing number.

Syntax	`void setRoutingNumber (String` *value*`);`
Parameters	*value* — A string that specifies the routing number
Discussion	This method only changes the value of the routing number field in the object; it does not modify the value in the database.

setType()

Specifies the financial account's payment type.

Syntax	`void setType (String` *value*`);`
Parameters	*value* — A string that specifies the type.
Discussion	This method only changes the value of the type field in the object; it does not modify the value in the database.

BXMembershipContext Class Reference

The BXMembershipContext class implements the IBXDBContext interface. Instances of the BXMembershipContext class represent a database context for membership objects.

You typically create a membership object using one of the BXMembershipMgr class methods and then create a context for it by creating a BXMembershipContext object. Because membership objects are not stored in a database per se, the transaction methods are written to throw an exception if called. Because there is no database connection to close, the `release()` method performs no actions.

Class Definition

Table 3-10 describes the BXMembershipContext class definition.

Table 3-10 BXMembershipContext Class Definition

Package	`com.iplanet.ecomm.billxb2b.ejb.membership;`
Syntax	`public class BXMembershipContext implements IBXDBContent {...}`

Log Message Level Constants

The following constants are used to specify the level associated with log messages for BXMembershipContext class.

ejbAssociateUser()

Returns associates an account with a user.

Syntax	`public static final int ERROR = -1;`
	`public static final int WARNING = 0;`
	`public static final int INFO = 1;`

ERROR	Error message
WARNING	Warning message
INFO	Informational message

Constructor

This section describes the BXMembershipContext constructor.

Syntax

```
BXMembershipContext ( IContext context ) throws Exception;
BXMembershipContext ( IContext context, BXStatus status ) throws
Exception;
```

Parameters

context — An IContext object that specifies the context.

status — A BXStatus object that holds the current status of the contex

Discussion

Creates a BXMembershipContext object. You typically create a membership context object by specifying the applogic's *context* variable for the *context* parameter. The first form of the constructor does not associate a *status* object with the *context*. In this case, you must call the `setStatus()` or `getBXStatus()` method to specify a status object for the context.

Methods

This section describes the BXMembershipContext methods. Table 3-11 lists these methods.

Table 3-11 BXMembershipContext Methods *(1 of 2)*

Method	Description
Constructor	
`BXMembershipContext()`	Creates a BXMembershipContext object.
Getting and setting the status	
`getStatus()`	Determines the status object currently associated with this entity.
`getBXStatus()`	Determines the status object currently associated with this object.
`setStatus()`	Specifies the status object to associate with this object.
Terminating the context	
`release()`	Closes the database connection and releases memory for this context object.
`finalize()`	Cleans up context and releases resources.
Transaction Management	
`beginTransaction()`	Starts a transaction.
`commitTransaction()`	Commits a transaction.
`rollbackTransaction()`	Rolls back a transaction.

Table 3-11 BXMembershipContext Methods *(2 of 2)*

Method	Description
Debugging Support	
`log()`	Writes a message to the log.

beginTransaction()

Starts a transaction.

Syntax	`void beginTransaction() throws Exception;`
Parameters	None.
Discussion	The `beginTransaction()` method is not supported for membership objects. This method raises an exception if it is called.

commitTransaction()

Commits a transaction.

Syntax	`void commitTransaction() throws Exception;`
Parameters	None.
Discussion	Not supported for membership objects. This method raises an exception if it is called.

finalize()

Cleans up context and releases resources.

Syntax	void finalize();
Parameters	None.
Discussion	The `finalize()` method is called automatically when the object is garbage collected; you do not call this method directly. The `finalize()` method calls the object's `release()` method.

getBXStatus()

Determines the status object currently associated with this object.

Syntax	`BXStatus getBXStatus();`
Parameters	None.
Discussion	If a BXStatus object has not been associated with the context, the `getBXStatus()` method creates a new BXStatus object and returns a reference to it. The `getBXStatus()` method is a protected method. Returns a BXStatus object that specifies the current status

getStatus()

Determines the status object currently associated with this entity.

Syntax	`BXStatus getStatus();`
Parameters	None.
Discussion	Returns a BXStatus object that specifies the current status.

Log()

Writes a message to the log.

Syntax	`void log( int` *severity*`, String` *msg*`);`
Parameters	*severity* — An integer that specifies the message level. *msg* — A string that contains the message.
Discussion	Use ERROR, WARNING, or INFO for the severity parameter.

release()

Cleans up context and releases resources.

Syntax	`void release();`
Parameters	None.
Discussion	This method does not perform any action.

rollbackTransaction()

Rolls back a transaction.

Syntax	`void rollbackTransaction() throws Exception;`
Parameters	None.
Discussion	The `rollbackTransaction()` method is not supported for membership objects. This method raises an exception if it is called.

setStatus()

Specifies the status object to associate with this object.

Syntax	`BXStatus setStatus (BXStatus` *status*`);`
Parameters	*status* — A BXStatus object.
Discussion	Returns a reference to the status object.

BXMembershipEntity Class Reference

The BXMembershipEntity class is an implementation of the IBXMembershipEntity interface, which is the base interface for all membership objects.

Whenever possible, use the interface class IBXMembershipEntity (see “IBXMembershipEntity Class Reference”) instead of BXMembershipEntity.

IBXMembershipEntity Class Reference

The IBXMembershipEntity class is the base interface for all membership objects. You must call the `setMembershipContext()` method of each object that represents Returns a membership record in the database before you can manipulate the record with the object.

Interface Definition

Table 3-12 describes the IBXMembershipEntity interface definition.

Table 3-12 IBXMembershipEntity Interface Definition

Package	`com.iplanet.ecomm.billxb2b.ejb.membership;`
Syntax	`public interface IBXMembershipEntity extend IBXEntity {...}`

Methods

This section describes the IBXMembershipEntity methods. Table 3-13 lists these methods.

Table 3-13 IBXMembershipEntity Interface Methods

Methods	Description
`getMembershipContext()`	Determines the membership context.
`setMembershipContext()`	Specifies the membership context.

getMembershipContext()

Determines the membership context.

Syntax	`BXMembershipContext getMembershipContext();`
Parameters	None.
Discussion	The BXMembershipContext object associated with this membership object.

setMembershipContext()

Specifies the membership context.

Syntax	`void setMembershipContext (BXMembershipContext` *context*`);`
Parameters	*context* — The context associated with this membership object.
Discussion	

BXMembershipMgr Class Reference

The BXMembershipMgr class creates membership-related business objects. You typically create a membership object and use it to manipulate a record in a Directory Server database.

Class Definition

Table 3-14 describes the BXMembershipMgr class definition.

Table 3-14 BXMembershipMgr Class Definition

Package	`com.iplanet.ecomm.billxb2b.ejb.membership;`
Syntax	`public class BXMembershipMgr {...}`

Methods

This section describes the BXMembershipMgr class methods. Table 3-15 lists these methods.

Table 3-15 BXMembershipMgr Class Methods

Methods	Description
Create membership object	
`createAccountType()`	Creates an account type object.
`createAddress()`	Creates an address object.
`createBillerProfile()`	Creates a billing profile object.
`createBillerGroup()`	Creates a biller group object
`createCustomerProfile()`	Creates a customer profile object.
`createCompanyProfile()`	Creates a company profile object.
`createDepartment()`	Creates a department object.
`createFinancialAccount()`	Creates a financial account object.
`createGroup()`	Creates a group object.
`createPaymentType()`	Creates a payment type object.
Create Entity Object	
`createObject()`	Creates an object.

createAccountType()

Creates an account type object.

Syntax

```
static IBXAccountType createAccountType (String billName ) throws
Exception;
```

Parameters *billName* — A string that contains the object mapping key in the format of billerName:acctType.

Discussion The `createAccountType()` method is a class method. If you use only the default account, specify null for the *billName* parameter. Returns an IBXAccountType object.

createAddress()

Creates an address object.

Syntax

```
static IBXAccountType createAccountType (String billName )
throws Exception;
```

Parameters *billName* — A string that contains the object mapping key in the format of billerName:acctType.

Discussion The `createAccountType()` method is a class method. If you use only the default account, specify null for the *billName* parameter. Returns an IBXAccountType object.

createBillerProfile()

Creates a billing profile object.

Syntax

```
static IBXBillerProfile createAccountType (String billName ) throws
Exception;
```

Parameters *billName* — A string that contains the object mapping key in the format of billerName:acctType.

Discussion The `createBillerProfile()` method is a class method. If you use only the default account, specify null for the *billName* parameter. Returns an IBXAccountType object.

createBillerGroup()

Creates a Biller group object.

Syntax

```
static IBXBillerGroup createBillerGroup (String billName )
throws Exception;
```

Parameters	*billName* — A string that contains the object mapping key in the format of billerName:acctType.
Discussion	Returns an IBXBillerGroup object.

createCompanyProfile()

Creates a company profile object.

Syntax	`static IBXCompanyProfile createCompanyProfile (String` *companyName* `) throws Exception;`
Parameters	*companyName* — A string that contains the company name
Discussion	Returns an IBXCompanyProfile object.

createCustomerProfile()

Creates a customer profile object.

Syntax	`static IBXCustomerProfile createCustomerProfile (String` *billName* `) throws Exception;`
Parameters	*billName* — A string that contains the object mapping key in the format of billerName:acctType.
Discussion	The `createAccountType()` method is a class method. If you use only the default account, specify null for the *billName* parameter. Returns an IBXCustomerProfile object.

createDepartment()

Creates a department object.

Syntax	`static IBXDepartment createDepartment( String` *departmentName* `) throws Exception;`
Parameters	*departmentName* — A string that contains the deparment name
Discussion	Returns an IBXDeparment object.

createFinancialAccount()

Creates a financial account object.

Syntax

```
static IBXFinancialAccount createFinancialAccount( String billName ) throws Exception;
```

Parameters

billName — A string that contains the object mapping key in the format of billerName:acctType.

Discussion

The `createFinancialAccount()` method is a class method. If you use only the default account, specify null for the *billName* parameter. Returns an IBXFinancialAccount object.

createGroup()

Creates a group object.

Syntax

```
static IBXGroup createGroup( String billName ) throws Exception;
```

Parameters

billName — A string that contains the object mapping key in the format of billerName:acctType.

Discussion

Returns an IBXGroup object.

createObject()

Creates an object.

Syntax

```
static IBXEntity createObject( String billName ) throws Exception;
```

Parameters

className — A string that specifies the class to which the object belongs

billName — A string that contains the object mapping key in the format of billerName:acctType.

Discussion

The `createObject()` method is a protected class method. If you use only the default account, specify null for the *billName* parameter. Returns an IBXEntity object.

createPaymentType()

Creates a payment type object.

Syntax	`static IBXPaymentType createPaymentType( String` *billName* `) throws Exception;`
Parameters	*billName* — A string that contains the object mapping key in the format of billerName:acctType
Discussion	The `createPaymentType()` method is a class method. If you use only the default account, specify null for the *billName* parameter. Returns an IBXPaymentType object.

BXPaymentType Class Reference

The BXPaymentType class is an implementation of the IBXPaymentType interface, which defines customer profile objects that represent payment type records.

Whenever possible, use the interface class IBXPaymentType (see "IBXPaymentType Class Reference") instead of BXPaymentType.

IBXPaymentType Class Reference

The IBXPaymentType class interface defines customer profile objects that represent payment type records. You call the `BXMembershipMgr.createPaymentType()` method to create a payment type object.

Before you can call the `IBXEntity.ejbCreate()` method to create a new record Returns associated with the object, you must call the `setType()` method to set the type field.

Interface Definition

Table 3-14 describes the IBXPaymentType class definition.

Table 3-16 BXMembershipMgr Class Definition

Package	`com.iplanet.ecomm.billxb2b.ejb.membership;`
Syntax	`public class BXMembershipMgr {...}`

Methods

This section describes the IBXPaymentType interface methods.

Table 3-17 IBXPaymentType Interface Methods

Method	Description
Getting and setting payment type information	
`getType()`	Determines the payment type.
`setType()`	Specifies the payment type.
`getBrand()`	Determines the card's brand.
`setBrand()`	Specifies the card's brand.
`getAlias()`	Determines the user-defined name associated with this payment type.
`setAlias()`	Specifies the user-defined name associated with this payment type.
`getDescription()`	Determines the payment type's description.
`setDescription()`	Specifies the payment type's description.
`getMinAmount()`	Determines the minimum payment.
`setMinAmount()`	Specifies the minimum payment.
`getMaxAmount()`	Determines the maximum payment.
`setMaxAmount()`	Specifies the maximum payment.
`getProcessor()`	Determines the credit card processor.
`setProcessor()`	Specifies the credit card processor.
`getPaymentStatus()`	Determines the payment status
`setPaymentStatus()`	Specifies the payment status

getAlias()

Determines the user-defined name associated with this payment type.

Syntax	`String getAlias();`
Parameters	None.
Discussion	Returns a string that specifies the alias.

getBrand()

Determines the card's brand.

Syntax	`String getBrand();`
Parameters	None.
Discussion	Brands include: VISA, MasterCard, American Express, Discover, and so forth. Returns a string that specifies the brand.

getDescription()

Determines the payment type's description.

Syntax	`String getDescription();`
Parameters	None.
Discussion	Returns a string that specifies the description.

getMaxAmount()

Determines the maximum payment.

Syntax	`double getMaxAmount();`
Parameters	None.
Discussion	Returns a double precision floating point number that specifies the amount.

getMinAmount()

Determines the minimum payment.

Syntax	`double getMinAmount();`
Parameters	None.
Discussion	Returns a double precision floating point number that specifies the amount.

getPaymentStatus()

Determines the payment status.

Syntax	`String getPaymentStatus();`
Parameters	None.
Discussion	Returns a string that contains the payment status

getProcessor()

Determines the credit card processor.

Syntax	`String getProcessor();`
Parameters	None.
Discussion	Returns a string that specifies the name of the processor.

getType()

Determines the payment type.

Syntax	`String getType();`
Parameters	None.
Discussion	Returns a string that contains the payment type.

setAlias()

Determines the user-defined name associated with this payment type.

Syntax	`void setAlias (String` *value*`);`
Parameters	*value* — A string that specifies the alias.
Discussion	The method only changes the value of the alias field in the object, it does not modify the value in the database.

setBrand()

Specifies the card's brand.

Syntax	`void setBrand (String` *value*`);`
Parameters	*value* — A string that specifies the brand
Discussion	The method only changes the value of the brand field in the object, it does not modify the value in the database.

setDescription()

Specifies the payment type's description.

Syntax	`void setDescription (String` *value*`);`
Parameters	*value* — A string that specifies the description
Discussion	The method only changes the value of the description field in the object, it does not modify the value in the database.

setMaxAmount()

Specifies the maximum payment.

Syntax	`void setMaxAmount (double` *value*`);`
Parameters	*value* — A double precision floating point number that specifies the amount
Discussion	The method only changes the value of the maximum amount field in the object, it does not modify the value in the database.

setMinAmount()

Specifies the minimum payment.

Syntax	`void setMinAmount (double` *value*`);`
Parameters	*value* — A double precision floating point number that specifies the amount.
Discussion	The method only changes the value of the value field in the object, it does not modify the value in the database.

setPaymentStatus()

Specifies the payment status.

Syntax	`void setPaymentStatus (String` *value*`);`
Parameters	*value* — A string that specifies the payment status
Discussion	The method only changes the value of the payment status field in the object, it does not modify the value in the database.

setProcessor()

Specifies the credit card processor.

Syntax	`void setProcessor (String` *value*`);`
Parameters	*value* — A string that specifies the value
Discussion	The method only changes the value of the field in the object, it does not modify the value in the database.

setType()

Specifies the payment type.

Syntax	`void setType (String` *value*`);`
Parameters	*value* — A string that specifies the payment type
Discussion	The method only changes the value of the type field in the object, it does not modify the value in the database.

Chapter 4

Event Component Business Object Interfaces

This chapter discusses component business object interface classes for events. These classes provide events and event handlers for handling asynchronous events within BillerXpert.

This chapter includes the following sections:

- Overview
- BXEvent Class Reference Descriptions

Overview

Event classes represent event records in the database. The event server uses these classes to manipulate event-related objects. You can use these classes to get and set fields in the objects. Table 4-1 lists these classes.

Table 4-1 Event Classes *(1 of 2)*

Class Name	Description	Location
1. BXEvent	Base class that manipulates event objects.	page 224
2. IBXEvent	Interface class that manipulates event objects.	page 224
3. BXEventContext	Establishes a context between an AppLogic and a billing component business object.	page 229
4. BXEventEntity	Base class that manipulates event-related objects.	page 230
5. IBXEventEntity	Interface class that manipulates event-related objects.	page 231
6. BXEventHandler	Base class that manipulates event handler mapping objects.	page 232

Table 4-1 Event Classes *(2 of 2)*

Class Name	Description	Location
7. IBXEventHandler	Interface class that manipulates event handler mapping objects.	page 232
8. BXEventMgr	Creates event-related component business objects.	page 238
9. BXEventQueue	Base class that manipulates event queue objects.	page 241
10. IBXEventQueue	Interface class that manipulates event queue objects.	page 241
11. BXHandler	Base class that manipulates handler objects.	page 250
12. IBXHandler	Interface class that manipulates handler objects.	page 250

BXEvent Class Reference

The BXEvent base class implements the IBXEvent interface, which defines objects that represent event records.

Whenever possible, use the interface class IBXEvent (see “IBXEvent Class Reference) instead of BXEvent.

IBXEvent Class Reference

The IBXEvent interface defines methods that manipulate objects representing event records. To create an event object, call the BXEventMgr.createEvent() method (see page 239) to create an event object.

Before you can call the `IBXEntity.IBXEntity.ejbCreate()` method (see page 129) to create a new record associated with the object, you must call the `setEventName()` method (see page 228) to set the event name.

Interface Definition

Table 4-2 describes the IBXEvent interface definition.

Table 4-2 IBXEvent Class Definition

Package	`com.iplanet.ecomm.billxb2b.ejb.event;`
Syntax	`public interface IBXEvent extends IBXEventEntity {...}`

Methods

This section describes the IBXEvent class methods. Table 4-3 lists these methods.

Table 4-3 IBXEvent Class Methods

Methods	Description
Manipulating customer account and related records	
`ejbFindAll()`	
`ejbFindByEventName()`	Retrieves an event record by event name.
`ejbRemoveAll()`	Removes all specified events. Getting and setting event information
`getEventDesc()`	Determines the event description.
`setEventDesc()`	Specifies the event description.
`getEventID()`	Determines the event ID.
`setEventID()`	Specifies the event ID.
`getEventName()`	Determines the event name.
`setEventName()`	Specifies the event name
`getBillerID()`	Determines the Biller ID.
`setBillerID()`	Specifies the Biller ID
`getEventStatus()`	Determines the event status.
`setEventStatus()`	Specifies the event status

ejbFindAll()

Retrieves the biller name.

Syntax `public SearchResultSet ejbFindAll (BXFilter` *filter*`, BXOrder` *order*`, int` *startIndex*`, int` *length*`, int` *maxRows*`) throws Exception;`

Parameters

filter — A BXFilter object that specifies the filter.

order — A BXOrder object that specifies the search order.

startindex — An integer that indicates where the search starts.

length — An integer that indicates the length.

maxRows — An integer that indicates the maximum number of rows.

Discussion

ejbFindByEventName()

Retrieves an event record by event name.

Syntax	`public void ejbFindByEventName (String` *EventName*`) throws Exception;`
Parameters	*EventName* — A string that specifies the event name.
Discussion	Retrieves the event record specified by the event name. The event's database context must previously have been set by calling this object's `setEventContext()` method. The values of fields already set in the object are overwritten by this method; values in the database are not affected.

ejbRemoveAll()

Removes all specified events.

Syntax	`public void ejbRemoveAll (BXFilter` *filter*`) throws Exception;`
Parameters	*filter* — A BXFilter object that specifies the filter.
Discussion	Removes the event records specified by the filter parameter; if null is specified, all events are deleted. The event's database context must previously have been set by calling this object's `setEventContext()` method. The `ejbRemoveAll()` method can be committed or rolled back as part of a transaction.

getEventDesc()

Determines the event description.

Syntax	`public String getEventDesc ();`
Parameters	None.
Discussion	Returns a string that contains the description.

getEventID()

Determines the event ID.

Syntax	`public String getEventID ();`

Parameters	None.
Discussion	Returns a string that contains the ID.

getEventID()

Determines the event name.

Syntax	`public String getEventID();`
Parameters	None.
Discussion	Returns a string that contains the event ID.

getEventName()

Determines the event name.

Syntax	`public String getEventName ();`
Parameters	None.
Discussion	Returns a string that contains the name.

getBillerID()

Determines the event name.

Syntax	`public String getBillerID ();`
Parameters	None.
Discussion	Returns a string that contains the Biller ID.

getEventStatus()

Determines the event name.

Syntax	`public String getEventStatus ();`
Parameters	
Discussion	Returns a string that contains the event status.

setEventDesc()

Specifies the event description.

Syntax	`public void setEventDesc (String` *eventDesc*`)`
Parameters	*eventDesc* — A string that specifies the description.
Discussion	This method only changes the value of the event description in the object; it does not modify the value in the database.

setEventID()

Specifies the event ID.

Syntax	`public void setEventID (String` *eventID*`)`
Parameters	*eventID* — A string that specifies the ID.
Discussion	This method only changes the value of the event ID in the object; it does not modify the value in the database.

setEventName()

Determines the event name.

Syntax	`public void setEventName (String` *eventName*`)`
Parameters	*eventName* — A string that specifies the name
Discussion	This method only changes the value of the event name in the object; it does not modify the value in the database.

setBillerID()

Determines the event name.

Syntax	`public void setBiller ID (String` *billerID*`)`
Parameters	*billerID* — A string that specifies the biller ID
Discussion	This method only changes the value of the biller ID in the object; it does not modify the value in the database.

setEventStatus()

Determines the event name.

Syntax	`public void setEventStatus (String` *status*`)`
Parameters	*status* — A string that specifies the status
Discussion	This method only changes the value of the status in the object; it does not modify the value in the database.

BXEventContext Class Reference

The BXEventContext class implements the IBXDBContext interface. Instances of the BXEventContext class represent a database context for event objects. You typically create an object using one of the BXEventMgr class methods and then create a context for it by creating a BXEventContext object.

Class Definition

Table 4-4 describes the BXEventContext class definition.

Table 4-4 BXEventContext Class Description

Package	`com.iplanet.ecomm.billxb2b.ejb.event`
Syntax	`public class BXEventContext extends BXDBContext {...}`

Constructors

The BXEvent Class contains the following constructor:

Syntax	`public BXEventContext {IContext` *`context`*`} throws Exception;`
	`public BXEventContext {IContext` *`context`*`, BXStatus` *`status`*`} throws Exception;`
	`public BXEventContext {IContext` *`context`*`,`
	`String` *`datasource`*`,`
	`String` *`database`*`,`
	`String` *`username`*`,`
	`String` *`passwd`*`,`
	`int` *`trace`*`};`

Parameters	*context* — An IContext object that specifies the context. *status* — A BXStatus object that holds the current status of the context *datasource* — A string that specifies the data source *database* — A string that specifies the database *username* — A string that specifies the user name *passwd* — A string that specifies the password.
Discussion	Creates a BXEventContext object. You typically create an event context object by specifying the applogic's *context* variable for the *context* parameter. The first and third forms of the constructor do not associate a status object with the context. In this case, you must call the `setStatus()` or `getBXStatus()` method to specify a status object for the context. The first two forms of the constructor take the database login information from a configuration file. If you use the third form of the constructor, specify the same value for the *datasource* parameter as you do for the *database* parameter.

BXEventEntity Class Reference

The BXEventEntity class is an implementation of IBXEventEntity interface, which is the base interface for all event objects.

Whenever possible, use the interface class IBXEventEntity (see "BXEventHandler Class Reference") instead of BXAddress.

IBXEventEntity Class Reference

The IBXEventEntity class is the base interface for all event objects. You must call the `setEventContext()` method of each object that represents a billing record in the database before you can manipulate the record with the object.

Interface Definition

Table 4-5 describes the IBXEventEntity interface description.

Table 4-5 IBXEventEntity Interface Definition

Package	`com.iplanet.ecomm.billxb2b.ejb.event`
Syntax	`public interface IBXEventEntity extends IBXEntity {...}`

Methods

This section describes the IBXEventEntity interface methods. Table 4-6 lists these methods.

Table 4-6 IBXEventEntity Class Methods

Methods	Description
Getting and setting event context information	
`getEventContext()`	Determines the event context.
`setEventContext()`	Specifies the event context.

getEventContext()

Determines the event context.

Syntax	`public BXEventContext getEventContext ();`
Parameters	None.
Discussion	Returns the BXEventContext object associated with this event object.

setBillingContext()

Determines the event context.

Syntax	`public void setEventContext (BXEventContext` *context*`);`
Parameters	*context* — The context associated with this billing object
Discussion	

BXEventHandler Class Reference

The BXEventHandler class is an implementation of IBXEventHandler interface, which defines objects that represent event handler records.

Whenever possible, use the interface class IBXEventHandler (see "IBXEventHandler Class Reference") instead of BXEventHandler.

IBXEventHandler Class Reference

The IBXEventHandler class defines objects that represent event handler records. You call the `BXEventMgr.createEventHandler()` method to create an event handler object.

Before you can call the `IBXEntity.ejbCreate()` method to create a new record associated with the object, you must call the `setEventName()`, `setEventHandler()`, and `setEventHandlerStatus()` methods to set the event name, event handler name, and event handler status, respectively.

Class Description

Table 4-7 describes the IBXEventHandler class description.

Table 4-7 IBXEventHandler Class Description

Package	`com.iplanet.ecomm.billxb2b.ejb.event`
Syntax	`public interface IBXEvent extends IBXEventEntity {...}`

Methods

Table 4-8 describes the IBXEventHandler class methods.

Table 4-8 IBXEventHandler Class Methods

Methods	Description
Manipulating customer account and related records	
`ejbFindByEventName()`	Retrieves an event handler record by event name.
`ejbRemoveAll()`	Removes all specified event handlers.
Getting and setting credit card information	
`getBillerID()`	Determines the biller ID associated with this handler.
`setBillerID()`	Specifies the biller ID associated with this handler.
`getEventHandler()`	Determines the event handler's class name.
`setEventHandler()`	Specifies the event handler's class name.
`getEventHandlerID()`	Determines the event handler ID.
`setEventHandlerID()`	Specifies the event handler ID.
`getEventName()`	Determines the event name.
`setEventName()`	Specifies the event name.
`getExtension1()`	Determines extension 1
`setExtension1()`	Specifies extension 1
`getExtension 2()`	Determines extension 2
`setExtension 2()`	Specifies extension 2
`getExtension3()`	Determines extension 3
`setExtension3()`	Specifies extension 3
`getExtension4()`	Determines extension 4
`setExtension4()`	Specifies extension 4

ejbFindByEventName()

Retrieves an event handler record by event name.

Syntax

```
public void ejbFindByEventName (String EventName) throws Exception;
public void ejbFindByEventName (String EventName, String BillerID)
throws Exception;
```

Parameters

EventName — A string that specifies the event name

BillerID — A string that specifies the event name

Discussion

The `ejbFindByEventName()` method retrieves the event handler record specified by the event name. The fields of this object are set from the contents of the record. The event handler's database context must previously have been set by calling this object's `setEventContext()` method.

The values of fields already set in the object are overwritten by this method; values in the database are not affected.

ejbRemoveAll()

Removes all specified event handlers.

Syntax

```
public void ejbRemoveAll (BXFilter filter) throws Exception;
```

Parameters

filter — A BXFilter object that specifies the filter

Discussion

The `ejbRemoveAll()` method removes the event handler records specified by the filter parameter; if null is specified, all event handlers are deleted. The event handler's database context must previously have been set by calling this object's `setEventContext()` method. The `ejbRemoveAll()` method can be committed or rolled back as part of a transaction.

getBillerID

Determines the biller ID associated with this handler.

Syntax

```
public String getBillerID ();
```

Parameters

None.

Discussion

Returns a string that contains the ID.

getEventHandler()

Determines the event handler's class name.

Syntax	`public String getEventHandler ();`
Parameters	None.
Discussion	Returns a string that contains the handler's class name.

getEventHandlerStatus()

Determines the event handler status.

Syntax	`public String getEventHandlerStatus ();`
Parameters	None.
Discussion	Returns a string that contains the status.

getEventName()

Specifies the event name.

Syntax	`public String getEventName ();`
Parameters	None.
Discussion	Returns a string that contains the name.

getExtension1()

Specifies extension one.

Syntax	`public String getExtension1();`
Parameters	None.
Discussion	Returns a string that contains extension 1.

getExtension2()

Specifies extension one.

Syntax	`public String getExtension2();`
Parameters	None.
Discussion	Returns a string that contains extension 2.

getExtension3()

Specifies extension one.

Syntax	`public String getExtension3();`
Parameters	None.
Discussion	Returns a string that contains extension 3.

getExtension4()

Specifies extension one.

Syntax	`public String getExtension4 ();`
Parameters	None.
Discussion	Returns a string that contains extension 4.

setBillerID()

Specifies the biller ID associated with this handler.

Syntax	`public void setBillerID (String` *billerID*`);`
Parameters	*billerID* — A string that specifies the ID
Discussion	This method only changes the value of the event biller ID in the object; it does not modify the value in the database.

setEventHandler()

Specifies the event handler's class name.

Syntax	`public void setEventHandler (String` *eventHandler*`);`
Parameters	*eventHandler* — A string that specifies the event handler's name.
Discussion	This method only changes the value of the event handler in the object; it does not modify the value in the database.

setEventHandlerStatus()

Specifies the event handler status.

Syntax	`public void setEventHandlerStatus (String` *status*`);`
Parameters	*status* — A string that specifies the status
Discussion	This method only changes the value of the status in the object; it does not modify the value in the database.

setBillerName()

Specifies the biller name.

Syntax	`public void setBillerName (String` *value*`);`
Parameters	*value* — A string representing the biller name.
Discussion	

setEventName()

Specifies the event name.

Syntax	`public void setEventName (String` *eventName*`);`
Parameters	*eventName* — A string that specifies the event name .
Discussion	This method only changes the value of the event name in the object; it does not modify the value in the database.

setExtension1()

Specifies extension 1.

Syntax	`public void setExtension1 (String` *extension1*`);`
Parameters	*extension1* — A string that specifies extension 1.
Discussion	This method only changes the value of extension1 in the object; it does not modify the value in the database.

setExtension2()

Specifies extension 2.

Syntax	`public void setExtension2 (String` *extension2*`);`
Parameters	*extension2* — A string that specifies extension 1
Discussion	This method only changes the value of extension2 in the object; it does not modify the value in the database.

setExtension3()

Specifies extension 3.

Syntax	`public void setExtension3 (String` *extension3*`);`
Parameters	*extension3* — A string that specifies extension 1
Discussion	This method only changes the value of extensio3 in the object; it does not modify the value in the database.

setExtension4()

Specifies extension 4.

Syntax	`public void setExtension4 (String` *extension4*`);`
Parameters	*extension4* — A string that specifies extension 1
Discussion	This method only changes the value of extension4 in the object; it does not modify the value in the database.

BXEventMgr Class Reference

The BXBillingMgr class defines methods that create membership objects. You typically create a membership object and use it to manipulate a record in the database.

Class Definition

Table 4-9 describes the BXEventMgr class definition.

Table 4-9 BXEventMgr Class Definition

Package	`com.iplanet.ecomm.billxb2b.ejb.event`
Syntax	`public class BXEventMgr {...}`

Methods

Table 4-10 describes the BXEventMgr class methods.

Table 4-10 BXEventMgr Class Methods

Methods	Description
Create Billing objects	
`createEvent()`	Creates an event object.
`createEventHandler()`	Creates an event handler object.
`createEventQueue()`	Creates an event queue object.
`createHandler()`	Creates a handler object.
Create entity object	
`createObject()`	Creates an object.

createEvent()

Creates an event object.

Syntax	`public static IBXEvent createEvent (String` *billName*`) throws Exception;`
Parameters	*billName* — A string that contains the object mapping key in the format of billerName.acctType.

Discussion	The `createEvent()` method is a class method. If you use only the default account, specify null for the *billName* parameter. Returns an IBXEvent object;

createEventHandler()

Creates an event handler object.

Syntax	`public static IBXEventHandler createEventHandler (String` *billName*`) throws Exception;`
Parameters	*billName* — A string that contains the object mapping key in the format of billerName.acctType.
Discussion	The `createEventHandler()` method is a class method. If you use only the default account, specify null for the *billName* parameter. Returns an An IBXEventHandler object.

createEventQueue()

Creates an event queue object.

Syntax	`public static IBXEventQueue createEventHandler (String` *billName*`) throws Exception;`
Parameters	*billName* — A string that contains the object mapping key in the format of billerName.acctType.
Discussion	The `createEventQueue()` method is a class method. If you use only the default account, specify null for the billName parameter. Returns an IBXEventQueue object.

createHandler()

Creates a handler object.

Syntax	`public static IBXHandler createHandler (String` *billName*`) throws Exception;`
Parameters	*billName* — A string that contains the object mapping key in the format of billerName.acctType.
Discussion	The `createHandler()` method is a class method. If you use only the default account, specify null for the billName parameter. Returns an IBXHandler object.

createObject()

Creates an object.

Syntax	`static IBXEntity createObject (Sting` *`className`*`, String` *`billName`*`) throws Exception;`
Parameters	*className* — A string that specifies the class to which the object belongs *billName* — A string that contains the object mapping key in the format of billerName.acctType.
Discussion	The `createObject()` method is a protected class method. If you use only the default account, specify null for the *billName* parameter. Returns an IBEntity object.

BXEventQueue Class Reference

The BXEventQueue class is an implementation of IBXEventHandler interface, which defines objects that represent event queue records.

Whenever possible, use the interface class IBXEventHandler (see "IBXEventQueue Class Reference") instead of BXEventQueue.

IBXEventQueue Class Reference

The IBXEventQueue class defines objects that represents event queue records. You call the `BXEventMgr.createEventQueue()` method to create an event queue object.

Before you can call the `IBXEntity.ejbCreate()` method to create a new record associated with the object, you must call the `setEventName()` and `setEventData()` methods to set the event name and event data, respectively.

Interface Definition

Table 4-6 describes the IBXEventQueue interface definition.

Table 4-11 IBXEventQueue Interface Definition

Package	`com.iplanet.ecomm.billxb2b.ejb.event`
Syntax	`public interface IBXEventQueue extends IBXEventEntity {...}`

Methods

This section describes the IBXEventQueue interface methods. Table 4-12 lists these methods.

Table 4-12 IBXEventQueue Methods *(1 of 2)*

Methods	Description
Manipulating customer account and related records	
`ejbFindByEventID()`	Retrieves an event queue record by event ID.
`ejbXFindAll()`	Retrieves event queue records. Getting and setting event information
`ejbGetUniqueEventIDFromQueue()`	Determines an unique event ID from the event queue
`ejbGetEventIDsFromQueue()`	Determines event IDs from the event queue
Getting and setting event information	
`getBillerID()`	Determines the biller ID associated with this event queue entry.
`setBillerID()`	Specifies the biller ID associated with this handler.
`getEventData()`	Determines the event queue element's data.
`setEventData()`	pecifies the event queue element's data.
`getEventName()`	Determines the event name.
`setEventName()`	Specifies the event name.
`getEventPopDate()`	Determines the date the entry was logically removed from queue.
`setEventPopDate()`	Specifies the date the entry was logically removed from the queue.
`getEventPriority()`	Determines the event's priority.
`setEventPriority()`	Specifies the event's priority.
`getEventProcessDate()`	Determines the date the entry was processed.
`setEventProcessDate()`	Specifies the date the entry was processed.
`getEventPushDate()`	Determines the date the entry was added to the queue.
`setEventPushDate()`	Specifies the date the entry was added to the queue.
`getEventStatus()`	Determines the event queue entry's status.
`setEventStatus()`	Specifies the event queue entry's status.
`getEventQueueID()`	Determines the event queue ID.

Table 4-12 IBXEventQueue Methods *(2 of 2)*

Methods	Description
`setEventQueueID()`	Specifies the event queue ID.
`getEventDate()`	Determines the event date.
`setEventDate()`	Specifies the event date.
`getEventScheduleID()`	Determines the event schedule ID
`setEventScheduleID()`	Specifies the event schedule ID.

ejbFindByEventID()

Retrieves an event handler record by event name.

Syntax

```
public void ejbFindByEventName (String EventID) throws
Exception;
```

Parameters

EventID — A string that specifies the event ID.

Discussion

The `ejbFindByEventID()` method retrieves the event queue record specified by the *eventID*. The fields of this object are set from the contents of the record. The event queue object's database context must previously have been set by calling this object's `setEventContext()` method.

The values of fields already set in the object are overwritten by this method; values in the database are not affected.

ejbXFindAll()

Retrieves event queue records.

Syntax

```
public Vector ejbXFindAll (BXFilter filter, BXOrder order)
throws Exception;
```

Parameters

filter — A BXFilter that specifies the filter.

order — A string that specifies the order of the event queue records.

Discussion

A vector that contains `BXEventQueue` objects representing the selected records.

Discussion	The `ejbXFindAll()` method retrieves event queue records that have not been logically removed from the queue. The event queue object's database context must previously have been set by calling each object's `createEventContext()` method. For each record, the `ejbXFindAll()` method creates an object, sets the fields in the object from a retrieved record, and adds the object to the return vector.

ejbGetUniqueEventIDsFromQueue()

Retrieves unique event IDs from the event queue.

Syntax	`public void ejbGetUniqueEventIDsFromQueue() throws Exception;`
Parameters	None.
Discussion	The `ejbGetUniqueEventIDsFromQueue()` method retrieves the event queue record specified. The fields of this object are set from the contents of the record. The event queue object's database context must previously have been set by calling this object's `setEventContext()` method. The values of fields already set in the object are overwritten by this method; values in the database are not affected. Returns a vector that contains BXEventQueue objects representing the selected records.

ejbGetEventQueueIDsFromQueue()

Retrieves event queue IDs from the event queue.

Syntax	`public void ejbGetEventQueueIDsFromQueue() throws Exception;`
Parameters	None.
Discussion	The `ejbGetEventQueueIDsFromQueue()` method retrieves event IDs from the event queue. The fields of this object are set from the contents of the record. The event queue object's database context must previously have been set by calling this object's `setEventContext()` method. The values of fields already set in the object are overwritten by this method; values in the database are not affected. Returns a vector that contains BXEventQueue objects representing the selected records.

getBillerID()

Determines the biller ID associated with this event queue entry.

Syntax	`public String getBillerID();`
Parameters	None.
Discussion	Returns a string that contains the ID.

getEventData()

Determines the event queue element's data.

Syntax	`public String getEventData();`
Parameters	None.
Discussion	Returns a string that contains the name.

getEventPopDate()

Determines the date the entry was logically removed from the queue.

Syntax	`public Date getEventPopDate();`
Parameters	None.
Discussion	Returns a Date object that contains the date.

getEventPriority()

Determines the event's priority.

Syntax	`public String getEventName();`
Parameters	None.
Discussion	Returns a string that contains the priority.

getEventProcessDate()

Determines the date the entry was processed.

Syntax	`public Date getEventPopDate();`
Parameters	None.
Discussion	Returns a Date object that contains the date.

getEventPushDate()

Determines the date the entry was added to the queue.

Syntax	`public Date getEventPopDate();`
Parameters	None.
Discussion	Returns a Date object that contains the date.

getEventStatus()

Determines the event queue entry's status.

Syntax	`public String getEventStatus();`
Parameters	None.
Discussion	Returns a string that contains the status.

getEventQueueID()

Determines the event queue id.

Syntax	`public String getEventQueueID();`
Parameters	None.
Discussion	Returns a string that contains the event queue ID.

getEventDate()

Determines the event queue entry's status.

Syntax	`public Date getEventDate();`
Parameters	None.
Discussion	Returns a Date that contains the event's date.

getEventScheduleID()

Determines the event schedule ID.

Syntax	`public String getEventScheduleID();`
Parameters	None.
Discussion	Returns a string that contains the event schedule ID.

setBillerID()

Specifies the biller ID associated with this handler.

Syntax	`public void setBillerID (String` *billerID*`);`
Parameters	*billerID* — A string that specifies the ID.
Discussion	This method only changes the value of the event biller ID in the object; it does not modify the value in the database

setEventData()

Specifies the event queue element's data.

Syntax	`public void setEventData (String` *eventData*`);`
Parameters	*eventData* — A string that specifies the data
Discussion	This method only changes the value of the event data in the object; it does not modify the value in the database.

setEventID()

Specifies the event name.

Syntax	`public void setEventID (String` *eventID*`);`
Parameters	*eventID* — A string that specifies the event ID
Discussion	This method only changes the value of the event ID in the object; it does not modify the value in the database.

setEventPopDate()

Specifies the date the entry was logically removed from the queue.

Syntax	`public void setEventPopDate (Date` *popDate*`);`
Parameters	*popDate* — A Date object that specifies the date.
Discussion	This method only changes the value of the pop date in the object; it does not modify the value in the database.

setEventPriority()

Specifies the event's priority.

Syntax	`public void setEventPriority (String` *eventPriority*`);`
Parameters	*eventPriority* — A string that specifies the priority
Discussion	

setEventProcessDate()

Specifies the date the entry was processed.

Syntax	`public void setEventProcessDate (Date` *processDate*`);`
Parameters	*processDate* — A Date object that specifies the date
Discussion	This method only changes the value of the process date in the object; it does not modify the value in the database.

setEventPushDate()

Specifies the date the entry was added to the queue.

Syntax	`public void setEventPushDate (Date` *pushDate*`);`
Parameters	*pushDate* — A Date object that specifies the date.
Discussion	This method only changes the value of the push date in the object; it does not modify the value in the database.

setEventStatus()

Specifies the event queue entry's status.

Syntax	`public void setEventStatus (String` *status*`);`
Parameters	*status* — A string that specifies the status
Discussion	This method only changes the value of the status in the object; it does not modify the value in the database.

setEventQueueID()

Specifies the event queue ID.

Syntax	`public void setEventQueueID (String` *eventQueueID*`);`
Parameters	*eventQueueID* — A string that specifies the event queue ID
Discussion	This method only changes the value of the event queue ID in the object; it does not modify the value in the database.

setEventDate()

Specifies the event date.

Syntax	`public void setEventDate (Date` *date*`);`
Parameters	*date* — A Date object that specifies the event date
Discussion	This method only changes the value of the event date in the object; it does not modify the value in the database.

setEventScheduleID()

Specifies the event schedule ID.

Syntax	`public void setEventScheduleID (String` *eventScheduleID*`);`
Parameters	*eventScheduleID* — A string that specifies the event schedule ID
Discussion	This method only changes the value of the event schedule ID in the object; it does not modify the value in the database.

BXHandler Class Reference

The BXHandler class is an implementation of IBXEventHandler interface, which defines objects that represent handler records.

Whenever possible, use the interface class IBXEventHandler (see "IBXHandler Class Reference") instead of BXHandler.

IBXHandler Class Reference

The IBXHandler class defines objects that represent handler records. A handler record defines the applogic that responds to an event and other information related to the applogic's environment. This class also defines methods that allow you to locate the handler by name and to remove handlers. You call the `BXEventMgr.createHandler()` method to create a handler object.

Before you can call the I`BXEntity.ejbCreate()` method to create a new record associated with the object, you must call the `setHandlerName()` and `setApplicationName()` methods to set the handler name and application name, respectively.

Class Definition

Table 4-13 describes the IBXHandler class definition.

Table 4-13 IBXHandler Class Definition

Package	`com.iplanet.ecomm.billxb2b.ejb.event`
Syntax	`public interface IBXHandler extends IBXEventEntity {...}`

Methods

Table 4-14 describes the IBXHandler class methods.

Table 4-14 IBXHandler Class Methods

Methods	Description
Manipulating customer account and related records	
`ejbFindByHandlerName()`	Retrieves an event handler record by handler name.
`ejbRemoveAll()`	Removes all specified handlers.
Getting and setting handler information	
`getApplicationName()`	Determines the applogic ID associated with this handler..
`setApplicationName()`	Specifies the applogic ID associated with this handler.
`getHandlerClassPath()`	Determines the handling applogic's class path.
`setHandlerClassPath()`	Specifies the handling applogic's class path.
`getHandlerDesc()`	Determines the handler's description.
`setHandlerDesc()`	Specifies the handler's description.
`getHandlerID()`	Determines the handler ID.
`setHandlerID()`	Specifies the handler ID.
`getHandlerName()`	Determines the handler's name.
`setHandlerName()`	Specifies the handler's name.
`getHandlerPackage()`	Determines the handling applogic's package.
`setHandlerPackage()`	Specifies the handling applogic's package.

ejbFindByHandlerName()

Retrieves an event handler record by handler name.

Syntax

```
public void ejbFindByHandlerName (String HandlerName) throws
Exception;
```

Parameters

HandlerName — A string that specifies the handler name.

Discussion	The `ejbFindByHandlerName()` method retrieves the handler record specified by the handler name. The fields of this object are set from the contents of the record. The handler's database context must previously have been set by calling this object's `setEventContext()` method. The values of fields already set in the object are overwritten by this method; values in the database are not affected.

ejbRemoveAll()

Removes all specified handlers.

Syntax	`public void ejbRemoveAll (BXFilter` *filter*`) throws Exception;`
Parameters	*filter* — A BXFilter object that specifies the filter.
Discussion	The `ejbRemoveAll()` method removes the handler records specified by the filter parameter; if null is specified, all handlers are deleted. The handler's database context must previously have been set by calling this object's `setEventContext()` method. The `ejbRemoveAll()` method can be committed or rolled back as part of a transaction

getApplicationName()

Determines the applogic ID associated with this handler.

Syntax	`public String getApplicationName();`
Parameters	None.
Discussion	Returns a string that contains the ID

getHandlerClassPath()

Determines the handling AppLogic's class path.

Syntax	`public String getHandlerClassPath();`
Parameters	None.
Discussion	Returns a string that contains the class path.

getHandlerDesc()

Determines the handler's description.

Syntax	`public String getHandlerDesc();`
Parameters	None.
Discussion	Returns a string that contains the handler's description.

getHandlerID()

Determines the handler ID.

Syntax	`public String getHandlerID();`
Parameters	None.
Discussion	Returns a string that contains the ID

getHandlerName()

Determines the handler's name.

Syntax	`public String getHandlerName();`
Parameters	None.
Discussion	Returns a string that contains the name.

getHandlerPackage()

Determines the handling applogic's package.

Syntax	`public String getHandlerPackage();`
Parameters	None.
Discussion	Returns a string that contains the package name.

setApplicationName()

Specifies the AppLogic ID associated with this handler.

Syntax	`public void setApplicationName (String` *applicationName*`);`
Parameters	*applicationName* — A string that specifies the applogic ID.
Discussion	This method only changes the value of the applogic ID in the object; it does not modify the value in the database.

Part 2

BillerXpert Database Schemas

Chapter 5

Billing Schema

This chapter describes the structure of the Oracle tables BillerXpert uses to handle billing.

This chapter includes the following sections:

- Billing Schema
- Relationship Between Tables
- Listing of Tables
- Event Tables
- Activity Tracking Tables
- Document Tables
- Payment Related Tables
- Assignment Tables
- Other Tables

CAUTION BillerXpert provides an API that you should use to access the database instead of the schema. The database schema for BillerXpert is subject to change in future versions of BillerXpert. If you rely on the schema, you should consider the potential reimplementation effort that you could incur as the result of an upgrade to the database.

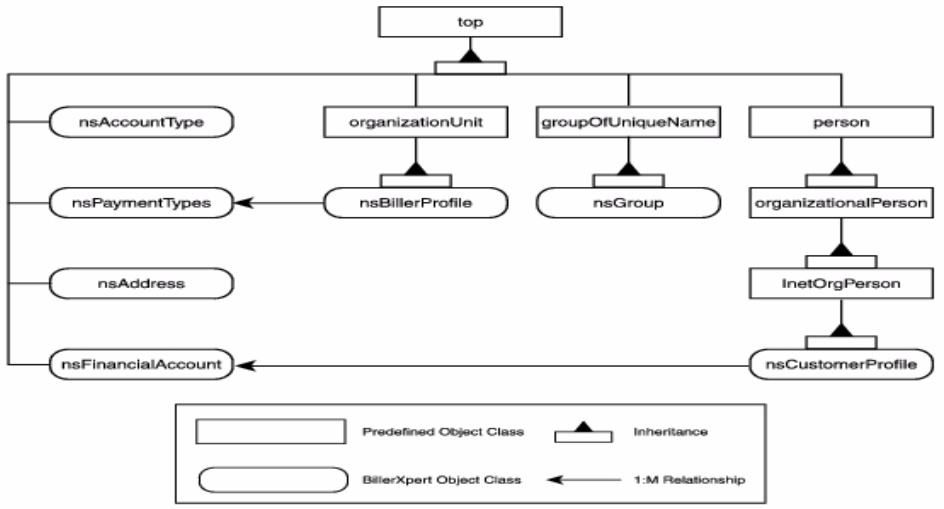
Billing Schema

The billing database schema consists of tables that represent customer account and statement information. It also includes tables on event and event handling information and relationship information between customers and accounts. The billing table listing provides a list of all the billing database schema tables and a short description of each table. Refer to a table's referenced page number for more information about that table.

Relationship Between Tables

Figure 5-1shows the relationship between the Billing tables.

Figure 5-1 Billing table relationships



Listing of Tables

Table 5-1 identifies the BillerXpert tables in the Oracle database.

Table 5-1 Oracle Database Billing Tables

Table Name	Functional Group	Description
1. BXBillerCustomerAccount	Document	BillerXpert customer account
2. BXContainer	Document	Containers
3. BXDocument	Document	Documents
4. BXSummary	Document	Summary lines
5. BXEvent	Events	Events
6. BXEventHandler	Events	Event handlers
7. BXEventQueue	Events	Event queue entries
8. BXHandler	Events	Handlers
9. BXEventLog	Activity Tracking	Event logs
10. BXEventLogDetail	Activity Tracking	Event log detail lines
11. BXEventSchedule	Activity Tracking	Schedule of Events
12. BXActivity	Activity Tracking	Activity logs
13. BXLoadTrack	Other	Track file loading.
14. BXMessage	Other	E-mail message information
15. BXError	Other	Error information
16. BXExtensionDesc	Other	Extension information
17. BXPayHistory	Payment	Payment histories
18. BXPaySchedule	Payment	Payment schedules
19. BXUserAcctAsgt	Assignment tables	Relationship between users and accounts
20. BXDocLineAsgt	Assignment tables	Relationship between document and line item details

Event Tables

This section describes the BillerXpert event tables.

BXEvent Table

The BXEvent table stores all BillerXpert event names. Table 5-2 lists this information.

Table 5-2 BXEvent Table

Name	Required	Type	Description
1. EventID	Yes	number(15)	Primary key — system generated.
2. EventName	Yes	varchar2(64)	Name of the event.
3. Description	No	varchar2(256)	Description of the event.
4. BillerID	Yes	varchar2(32)	Biller ID
5. Status	Yes	varchar2(32)	Status of event. Valid values are: • ACTIVE • INACTIVE • VOID
6. Extension1	No	varchar2(256)	Extension1 for events
7. Extension2	No	varchar2(256)	Extension2 for events
8. Extension3	No	varchar2(256)	Extension3 for events
9. Extension4	No	varchar2(256)	Extension4 for events

BXHandler Tables

The BXHandler table stores event handler information. Table 5-3 lists these handlers.

Table 5-3 BXHandler Table *(1 of 2)*

Name	Required	Type	Description
HandlerID	Yes	number(15)	Primary key — system generated.
HandlerName	Yes	varchar2(64)	Name of the event handler
ApplicationID	No	varchar2(64)	AppLogic name or GUI ID for the event handler.
Description	No	varchar2(256)	Description of the event handler.

Table 5-3 BXHandler Table *(2 of 2)*

Name	Required	Type	Description
Package	No	varchar2(256)	Used with the classpath field, represents the location of the class responsible for collecting configuration information for the handler. For example, for an email handler, these fields would collect address and subject information.
ClassPath	No	varchar2(256)	Used with the package field, represents the location of the class responsible for collecting configuration information for the handler. For example, for an e-mailhandler, these fields would collect address and subject information.
Extension1	No	varchar2(256)	Extension1 for handlers
Extension2	No	varchar2(256)	Extension2 for handlers
Extension3	No	varchar2(256)	Extension3 for handlers
Extension4	No	varchar2(256)	Extension4 for handlers

BXEventHandler Table

The BXEventHandler table stores the mapping information between events and event handlers. Table 5-4 lists this information.

Table 5-4 BXEventHandler Table *(1 of 2)*

Name	Required	Type	Description
EventHandlerID	Yes	number(15)	Primary key — system generated.
EventID	Yes	number(15)	Event ID — system generated
Status	Yes	varchar2(32)	Status of event to event handler mapping. Valid values are: • ACTIVE • INACTIVE • VOID
HandlerID	Yes	number(15)	Handler ID — system generated
Extension1	No	varchar2(256)	Extension1 for event handlers
Extension2	No	varchar2(256)	Extension2 for event handlers
Extension3	No	varchar2(256)	Extension3 for event handlers

Table 5-4 BXEventHandler Table *(2 of 2)*

Name	Required	Type	Description
Extension4	No	varchar2(256)	Extension4 for event handlers

BXEventQueue Table

The BXEventQueue table stores the event data. Table 5-5 lists this information.

Table 5-5 BXEventQueue Table *(1 of 2)*

Name	Required	Type	Description
EventQueueID	Yes	number(15)	Primary key — system generated.
BillerID	No	varchar2(64)	Biller ID.
EventID	Yes	number (15)	Event ID
Priority	No	number(15)	Not used.
Data	No	varchar2(2000)	Input data for the event. For example, in the case of an email event, where the event sends email to a user, the data might be: name=JMullen@netscape.com&BX_Template=/tmp/mail/template
Status	No	varchar2(32)	Status of the record. Values can either be “null” or “PROCESSING.” If a record is being processed, it cannot be processed by any other Event Server.
Pushed	No	date	Date when the event was pushed from the queue.
Popped	No	date	Date when the event was popped from the queue.
Processed	No	date	Date when the event was processed.
EventDate	Yes	date	Date when event is scheduled
EventScheduleID	No	number(15)	Event Schedule ID
Extension1	No	varchar2(256)	Extension1 for event queues
Extension2	No	varchar2(256)	Extension2 for event queues
Extension3	No	varchar2(256)	Extension3 for event queues

Table 5-5 BXEventQueue Table *(2 of 2)*

Name	Required	Type	Description
Extension4	No	varchar2(256)	Extension4 for event queues

Activity Tracking Tables

The activity tracking tables store activity tracking data.

BXEventLog Table

The BXEventLog table stores event log information. Table 5-6 lists this information.

Table 5-6 BXEventLog Table

Name	Required	Type	Description
EventLogID	Yes	number(15)	Primary key — system generated.
FileID	Yes	number(15)	File ID used by the loader.
CreateDate	yes	date	Date the event was create.
ApplName	Yes	varchar2(32)	Name of the AppLogic that initiated the activity.
Subject	Yes	varchar2(32)	Used with the Action field, identifies the activity associated with the event
Action	Yes	varchar2(32)	Used with the Subject field, identifies the activity associated with the event.
OrgID	No	varchar(64)	Biller ID.
AcctID	No	varchar(64)	Customer account ID.
CreatedBy	No	varchar2(64)	User name.
Message	No	varchar2(512)	Message associated with the activity.
Extension1	No	varchar2(256)	Extension1 for event queues
Extension2	No	varchar2(256)	Extension2 for event queues
Extension3	No	varchar2(256)	Extension3 for event queues
Extension4	No	varchar2(256)	Extension4 for event queues

BXEventLogDetail Table

The BXEventLog Detail Table table stores additional event activity tracking information that is not stored in the BXEventLog. Table 5-7 lists this information.

Table 5-7 BXEventLogDetail Table

Name	Required	Type	Description
EventLogDetailID	Yes	number(15)	Primary key — system generated.
EventLogID	Yes	number(15)	Primary key of BXEventLog table.
FileID	Yes	number(15)	Primary key of BXEventLog table.
Name	Yes	varchar2(32)	Used for additional information not included in the BX_Event_Log table.
Value	No	varchar2(256)	Used for additional information not included in the BX_Event_Log table.
Extension1	No	varchar2(256)	Extension1 for event log detail
Extension2	No	varchar2(256)	Extension2 for event log detail
Extension3	No	varchar2(256)	Extension3 for event log detail
Extension4	No	varchar2(256)	Extension4 for event log detail

BXEventSchedule Table

The BXEventSchedule table stores event scheduling activity information. Table 5-8 lists this information.

Table 5-8 BXEventSchedule Table *(1 of 2)*

Name	Required	Type	Description
EventScheduleID	Yes	number(15)	Primary key — system generated.
Description	No	varchar2(64)	Description of the event schedule
StartOnDate	Yes	date	Start date of event
StopOnDate	No	date	Stop date of event
ScheduleName	Yes	varchar2(64)	Name of scheduled event
SchMode	Yes	number(15)	Mode of scheduled event
DayOfMonth	No	number(15)	Scheduled day of the month event

Table 5-8 BXEventSchedule Table *(2 of 2)*

Name	Required	Type	Description
MonthFrequency	No	number(15)	Monthly frequency of the event
Mon	No	number(15)	Day of the week
Tue	No	number(15)	Day of the week
Wed	No	number(15)	Day of the week
Thu	No	number(15)	Day of the week
Fri	No	number(15)	Day of the week
Sat	No	number(15)	Day of the week
Sun	No	number(15)	Day of the week
LastProcessdDate	No	date	Date of last processed event
Extension1	No	varchar2(256)	Extension1 for event log detail
Extension2	No	varchar2(256)	Extension2 for event log detail
Extension3	No	varchar2(256)	Extension3 for event log detail
Extension4	No	varchar2(256)	Extension4 for event log detail

BXActivity Table

The BXActivity table stores event activity information. Table 5-9 lists this information.

Table 5-9 BXActivity Table

Name	Required	Type	Description
ActivityID	Yes	number(15)	Primary key — system generated.
ActivityTime	Yes	date	Time date of activity
UserID	Yes	varchar(128)	User ID of participant in activity
BillerID	Yes	varchar2(128)	Biller ID of activity
CustomerID	No	number(15)	Customer ID of activity
Activity	Yes	varchar(30)	Activity information
ActivityData	No	varchar2(2000)	Activity Data
EmulationID	No	varchar2(128)	Emulation ID
ActivityDetail	Yes	varchar2(2000)	Detail of activity

Document Tables

This section describes the BillerXpert document tables.

BXBillerCustomerAccount Table

The BXBillerCustomerAccount table stores the biller and customer account information. A single customer may have multiple accounts in this table. Biller / Customer will have a unique AcctID for each account in the BillerXpert database. The AcctID is used in conjunction with charge events, payment schedules, and log entries.

Table 5-10 lists this information.

Table 5-10 BXBillerCustomerAccount Table *(1 of 3)*

Name	Required	Type	Description
AcctID	Yes	number(15)	Customer account ID.
BillerID	Yes	varchar2(128)	Biller ID.
AccountType	No	varchar2(32)	Customer defined account type.
CustomerType	No	varchar2(32)	Customer type. The customer type can be: • REGULAR • SPECIAL
Description	No	varchar2(256)	Brief description of the Customer / Biller account. For example, “Account for residential address.”
TaxID	No	varchar2(64)	Biller / Customer tax identification number.
Since	No	number(8)	Date when the customer registered with the site.
ServiceOptions	No	varchar2(1024)	Available service options
Profile	No	varchar2(1024)	This tag represents the Customer / Biller profile information. For example: • AGE • PROFESSION • INCOME • SPECIAL INTERESTS
AccountNumber	Yes	varchar2(32)	Customer account number.

Table 5-10 BXBillerCustomerAccount Table *(2 of 3)*

Name	Required	Type	Description
AccountStatus	Yes	varchar2(32)	Status of an account. Valid statuses are: • ACTIVE • INACTIVE • PENDING • VOID
UnpaidBills	Yes	number(8)	Total number of bills left unpaid since the last bill date.
UnpaidBalance	Yes	decimal(16,4)	Total unpaid balance. This is a cumulative amount.
AccountBalance	Yes	decimal(16,4)	Current account balance.
AccountBalanceDate	Yes	date	Date when an account balance is reported.
CreateDate	Yes	date	Date on which the customer account was created.
ModifyDate	Yes	date	Date on which the customer account was modified.
DisputeStatus	Yes	varchar2(32)	Dispute status that can indicate any of the following: • If a dispute exists • If a dispute has been acted upon • If a dispute is closed Default values are: • OPEN • CLOSED • REOPEN • NODISPUTE
Extension	No	varchar2(2000)	Extra descriptive information related to the account.
BillingAddress	No	varchar2(64)	Customer's billing address.
ServiceAddress	No	varchar2(64)	Location of the service. This may also be the billing address.
VerifiedBy	No	varchar2(64)	Site implemented: name of the person who verifies account data entries.

Table 5-10 BXBillerCustomerAccount Table *(3 of 3)*

Name	Required	Type	Description
VerifiedDate	No	date	Site implemented: date on which the account data was verified.
AuthorizedBy	No	varchar2(64)	Site implemented: name of the authorizing person.
AuthorizedDate	No	date	Site implemented: date of authorization.
FileID	Yes	number(15)	File ID generated by the loader when it loads a file into the system.
AutoPayStatus	Yes	varchar(1)	Single character that contains pay status
Extension1	No	varchar2(256)	Extension1 for customer account detail
Extension2	No	varchar2(256)	Extension2 for customer account detail
Extension3	No	varchar2(256)	Extension3 for customer account detail
Extension4	No	varchar2(256)	Extension4 for customer account detail

BXContainer Table

This is the BillerXpert documentation container table. A container has one associated summary but can contain more than one document. Each document has its own associated summary.

Table 5-11 lists these containers.

Table 5-11 BXContainer Table *(1 of 2)*

Name	Required	Type	Description
ContainerID	Yes	number(15)	Primary key — system generated.
AcctID	Yes	number(15)	Primary key — system generated.
FileID	Yes	number(15)	Primary key — system generated
Description	No	varchar2(256)	Descriptive information about the container.
Header	No	varchar2(128) H	eading information related to the container.
Footer	No	varchar2(128)	Footer information related to the container.
Extension	No	varchar2(2000)	Additional descriptive information related to the container.

Table 5-11 BXContainer Table *(2 of 2)*

Name	Required	Type	Description
Extension1	No	varchar2(256)	Extension1 for document detail
Extension2	No	varchar2(256)	Extension2 for document detail
Extension3	No	varchar2(256)	Extension3 for document detail
Extension4	No	varchar2(256)	Extension4 for document detail

BXDocument Table

The BXDocument table is the billing document table. A document has one summary and may consist of many line items. Each document might be associated with a payment history.

Table 5-12 lists the billing document information.

Table 5-12 BXDocument Table *(1 of 2)*

Name	Required	Type	Description
DocID	Yes	number(15)	Primary key — system generated.
ContainerID	Yes	number(15)	Primary key — system generated.
AcctID	Yes	number(15)	Primary key — system generated.
FileID	Yes	number(15)	Primary key — system generated.
DocType	Yes	varchar2(32)	Document type. Valid document types are: • BILL • INVOICE • NOTICE • STATEMENT
BillType	Yes	varchar2(32)	Billing type. Valid billing types are: • TELCO • CABLE • UTILITY • BANKSTMT • MORTGAGESTMT
CreateDate	Yes	date	Document creation date.
ModifyDate	Yes	date	Document modification date.

Table 5-12 BXDocument Table *(2 of 2)*

Name	Required	Type	Description
FirstViewDate	No	date	Date on which the document was first viewed.
LastViewDate	No	date	Date on which the document was last viewed.
ViewCount	Yes	number(15)	Number of times the document was viewed. This is the difference between the FirstViewDate and the LastViewDate values.
Description	No	varchar2(256)	Description of the document.
BillDate	Yes	date	Date the bill was generated.
Status	Yes	varchar2(32)	Document status. Valid document statuses are: • ACTIVE • INACTIVE • VOID
CopyType	Yes	varchar2(64)	Document copy type. Valid document copy types are: • ORIGINAL • DUPLICATE
Extension	No	varchar(2000)	Additional descriptive information related to the document.
Extension1	No	varchar2(256)	Extension1 for document detail
Extension2	No	varchar2(256)	Extension2 for document detail
Extension3	No	varchar2(256)	Extension3 for document detail
Extension4	No	varchar2(256)	Extension4 for document detail

BXSummary Table

The BXSummary table is the billing summary table. It contains summary information about the customer's bill. Table 5-13 lists this information.

Table 5-13 BXSummary Table *(1 of 4)*

Name	Required	Type	Description
SumID	Yes	number(15)	Primary key — system generated.

Table 5-13 BXSummary Table *(2 of 4)*

Name	Required	Type	Description
ContainerID	Yes	number(15)	Primary key — system generated.
DocID	Yes	number(15)	Primary key — system generated.
FileID	Yes	number(15)	Primary key — system generated.
StartDate	Yes	date	Start of billing cycle date.
EndDate	Yes	date	End of the billing cycle date.
OriginalAmount	Yes	decimal(16,4)	Original amount of the bill. For example, a bill may be adjusted by a payment or other reason. The original amount is the amount before any of these adjustments.
CurrentBalance	Yes	decimal(16,4)	Current account balance.
CurrentCharge	Yes	decimal(16,4)	Current statement balance.
PreviousBalance	Yes	decimal(16,4)	Previous balance amount.
PreviousPayment	Yes	decimal(16,4)	Previous month's payment amount.
BalanceDate	Yes	date	Statement date.
DueDate	Yes	date	Statement due date.
LateAfterDate	No	date	Date after which the statement is considered late (that is, after the grace period).
PenaltyType	No	varchar2(64)	Penalty type. For example: • FIXED • PERCENT
PenaltyRate	No	decimal(16,4)	Penalty rate. For example: • If fixed, the penalty amount. • If ratio, the penalty percentage.
MinPayAmount	Yes	decimal(16,4)	Minimum payment amount.
PaymentAmount	Yes	decimal(16,4)	Previous payment amount received.
LastPayDate	No	date	Date the last payment was made.
PayStatus	Yes	varchar2(32)	Payment status. For example: • NOT_PAID • FULLY_PAID • PARTIALLY_PAID • OVER_PAID

Table 5-13 BXSummary Table *(3 of 4)*

Name	Required	Type	Description
DisputeAmount	Yes	decimal(16,4)	Amount of a bill or line item disputed by the customer.
DisputeCaseNum	No	varchar2(64)	Dispute identification number.
DisputeReasonCode	No v	archar2(64)	Dispute reason code. For example: • ALREADY_PAID • INVALID_ITEM • WRONGBILL • WRONGAMOUNT • WRONGITEM • OTHER
DisputeStatus	Yes	varchar2(32)	Dispute status. For example: • OPEN • CLOSED • REOPEN • NODISPUTE
Status	Yes	varchar2(32)	Billing summary status. For example: • ACTIVE • INACTIVE • VOID
PayTermsCode	No	varchar2(32)	For future use. This field can be used for passing along payment terms information from the legacy system.
DiscountBeforeDate	No	date	Date before which a discount will apply.
DiscountType	No	varchar2(32)	Discount type. For example: • FIXED • PERCENT
DicountRate	No	decimal(16,4)	Discount rate. For example: • If fixed, the amount of the discount. • If percent, the percentage of the discount.
Extension	No	varchar2(2000)	Additional descriptive information related to the billing summary.
URL	No	varchar2(256)	Site implemented: stores a URL, for example, to the detail bill.
Extension1	No	varchar2(256)	Extension1 for summary detail

Table 5-13 BXSummary Table *(4 of 4)*

Name	Required	Type	Description
Extension2	No	varchar2(256)	Extension2 for summary detail
Extension3	No	varchar2(256)	Extension3 for summary detail
Extension4	No	varchar2(256)	Extension4 for summary detail

Payment Related Tables

The Payment Related tables contain information about a customer's payment history and store information each time a customer pays a bill.

BXPayHistory Table

The BXPayHistory table contains a customer's payment history. Table 5-14 lists this information.

Table 5-14 BXPayHistory Table *(1 of 3)*

Name	Required	Type	Description
PayHistID	Yes	number(15)	Primary key — system generated.
AcctID	Yes	number(15)	Primary key — system generated.
DocID	Yes	number(15)	Primary key — system generated.
Type	Yes	varchar2(32)	Payment type. Valid values are credit card or check payment.
Brand	Yes	varchar2(32)	For credit cards: the type of credit card, for example, VISA.
PmtAmount	Yes	decimal(16,4)	Amount of payment.
CreateDate	Yes	date	Date the payment record was created in the system.
ModifyDate	Yes	date	Date the payment record was modified in the system.

Table 5-14 BXPayHistory Table *(2 of 3)*

Name	Required	Type	Description
PmtTxnStatus	Yes	varchar2(32)	Payment transaction status. Valid values are: • AUTH_PENDING • AUTH_SUCCESS • AUTH_FAIL • CAPTURE_PENDING • CAPTURE_IN_PROCESS • CAPTURE_SUCCESS • CAPTURE_FAIL • VOID_PENDING • VOID_SUCCESS • VOID_FAIL • REFUND_PENDING • REFUND_FAIL • NSF • SUBMITTED • CANCELLED • CHECK_SUB_PENDING • CHCK_SUB_IN_PROCESS • CHECK_SUB_SUCCESS • CHECK_SUB_FAILED • CHECK_VENDOR_FAILED • CHECK_VENDOR_PENDING • CHECK_VENDOR_SUCCESS • PENDING
PmtTxnID	Yes	number(15)	Transmission ID associated with the payment.
PmtMode	Yes	varchar2(32)	Payment mode. Valid values are: • AUTO • MANUAL
FileID	Yes	number(15)	File ID used by the loader.
SchedulePmtDate	Yes	date	Date the payment is scheduled.
NumProcessDays	No	number(15)	Number of process days for payment
ServerTrnID1	No	varchar2(1000)	Server Transaction ID1
ServerTrnID2	No	varchar2(1000)	Server Transaction ID2
PmtProcessDate	No	date	Date of payment processing.
Extension	No	varchar2(2000)	Used to store additional user-defined information.
Extension1	No	varchar2(256)	Extension1 for pay history detail

Table 5-14 BXPayHistory Table *(3 of 3)*

Name	Required	Type	Description
Extension2	No	varchar2(256)	Extension2 for pay history detail
Extension3	No	varchar2(256)	Extension3 for pay history detail
Extension4	No	varchar2(256)	Extension4 for pay history detail

BXPaySchedule Table

The BXPaySchedule table contains information related to automatic payment. This feature can be implemented by the site.

Table 5-15 lists this information.

Table 5-15 BXPaySchedule Table *(1 of 2)*

Name	Required	Type	Description
PaySchedID	Yes	number(15)	Primary key — system generated.
AcctID	Yes	number(15)	Primary key — system generated.
BillerID	Yes	varchar2(128)	Biller ID
BillerAcctNum	Yes	varchar2(128)	Biller Account Number
StartOnDate	No	date	Date on which the automatic payment starts.
StopOnDate	No	date	Date on which the automatic payment stops.
Frequency	No	varchar(32)	Frequency at which payments are automatically made.
PayMode	Yes	varchar(32)	Mode of payment . Values can be: • ONDUEDATE • BEFOREDUEDATE • ONFIXEDATE
DayBeforeDue	Yes	number(15)	Specifies n number of days to process the payment before the actual due date.
DayOfMonth	Yes	number(15)	Specifies day of the month payments
PayDate	No	date	Date of payment
PayAmount	No	decimal(16,4)	Amount of the payment.
PayAmountThreshold	No	decimal(16,4)	Amount threshold of the payment.

Table 5-15 BXPaySchedule Table *(2 of 2)*

Name	Required	Type	Description
PayAmountType	Yes	varchar2(64)	Payment amount type.Values can be: • FULL • FIXEDAMOUNT • MINIMUM • FIXEDPERCENT
PmtAcctID	Yes	varchar2(128)	Identifies the account from which to make a payment.
Status	Yes	varchar2(32)	Status. Valid values are: • ACTIVE • INACTIVE • VOID
Extension1	No	varchar2(256)	Extension1 for pay scheduling detail
Extension2	No	varchar2(256)	Extension2 for pay scheduling detail
Extension3	No	varchar2(256)	Extension3 for pay scheduling detail
Extension4	No	varchar2(256)	Extension4 for pay scheduling detail

Assignment Tables

This section describes the assignment tables.

BXUserAcctAsgt Table

The BXUserAcctAsgt table links the customer profile to the billing account. Table 5-16 lists this information.

Table 5-16 BXUserAcctAsgt Table

Name	Required	Type	Description
AcctID	Yes	number(15)	Biller Customer Account (BCA) ID.
UserID	Yes	varchar2(128)	Online user account login name.
Extension1	No	varchar2(256)	Extension1 for pay scheduling detail

Table 5-16 BXUserAcctAsgt Table

Name	Required	Type	Description
Extension2	No	varchar2(256)	Extension2 for pay scheduling detail
Extension3	No	varchar2(256)	Extension3 for pay scheduling detail
Extension4	No	varchar2(256)	Extension4 for pay scheduling detail

BXDocLineAsgt Table

The BXDocLineAsgt table links the document to the line item details. Table 5-17 lists this information.

Table 5-17 BXDocLineAsgt Table

Name	Required	Type	Description
DocID	Yes	number(15)	Primary key — system generated.
LineID	Yes	number(15)	Primary key — system generated.
FileID	Yes	number(15)	Primary key — system generated.
LineType	Yes	varchar2(32)	Interface to the detail table.
TableName	Yes	varchar2(32)	Line item's table name.
Extension1	No	varchar2(256)	Extension1 for line detail.
Extension2	No	varchar2(256)	Extension2 for line detail
Extension3	No	varchar2(256)	Extension3 for line detail.
Extension4	No	varchar2(256)	Extension4 for line detail.

Other Tables

This section lists other database tables that were not discussed previously.

BXLoadTrack Table

This table is used by the loading during the file loading process. Table 5-18 lists this information.

Table 5-18 BXLoadTrack Table

Name	Required	Type	Description
FileID	Yes	number(15)	Primary key — system generated.
BillerID	Yes	varchar2(128)	Biller ID.
Status	Yes	varchar2(16)	Status of the load procedure.
Path	Yes	varchar2(512)	Loader file path.
RecordCount	No	number(20)	Number of records inserted
FileType	No	varchar2(10)	Type of file inserted in the system.
LoadDate	No	date	Loading date
Extension1	No	varchar2(256)	Extension1 for load detail
Extension2	No	varchar2(256)	Extension2 for load detail
Extension3	No	varchar2(256)	Extension3 for load detail
Extension4	No	varchar2(256)	Extension4 for load detail

BXMessage Table

The BXMessage table is used to store email message information. Table 5-19 lists this information.

Table 5-19 BXMessage Table *(1 of 2)*

Name	Required	Type	Description
MessageID	Yes	number(15)	Primary key — system generated.
BillerID	Yes	varchar2(128)	Primary key — system generated.
AcctID	Yes	number(15)	Primary key — system generated.
DocID	Yes	number(15)	Primary key — system generated.
CreateDate	Yes	date	Date the message was created.
MsgTo	Yes	varchar2(32)	Destination address of the message.

Table 5-19 BXMessage Table *(2 of 2)*

Name	Required	Type	Description
MsgFrom	Yes	varchar2(32)	Sending address of the message.
Subject	Yes	varchar2(64)	Subject of the message.
Type	Yes	varchar2(32)	Type of message. Valid values are: • CORRESPONDENCE • DISPUTE
Message	Yes	varchar2(2000)	Content of the message.
SenderType	Yes	varchar2(32)	Sender type. Valid values are: • CUSTOMER • ADMIN • CSR
CaseNum	No	varchar2(32)	Case number for customer support reference.
CsrName	No	varchar2(64)	CSR name.
Status	Yes	varchar2(32)	Status. Valid values are: • ACTIVE • INACTIVE • VOID
Extension	No	varchar2(2000)	Used to store additional user-defined information.
Extension1	No	varchar2(256)	Extension1 for message detail
Extension2	No	varchar2(256)	Extension2 for message detail
Extension3	No	varchar2(256)	Extension3 for message detail
Extension4	No	varchar2(256)	Extension4 for message detail

BXExtensionDesc Table

The BXExtensionDesc table stores information about extensions. Table 5-20 lists this information.

Table 5-20 BXExtensionDesc Table

Name	Required	Type	Description
Tablename	Yes	varchar2(256)	Table extension name
Fieldname	Yes	varchar2(256)	Field extension name

Table 5-20 BXExtensionDesc Table

Name	Required	Type	Description
Description	Yes	varchar2(256)	Description extension
Extension1	No	varchar2(256)	Extension1 for detail
Extension2	No	varchar2(256)	Extension2 for detail
Extension3	No	varchar2(256)	Extension3 for detail
Extension4	No	varchar2(256)	Extension4 for detail

BXError Table

This BXError table is used to store error information. Table 5-21 lists this information.

Table 5-21 BXError Table

Name	Required	Type	Description
UserID	Yes	varchar2(128)	UserID name
EmulationID	No	varchar2(128)	Emulation ID name
BillerID	Yes	varchar2(128)	BillerID name
ErrorTime	Yes	date	Time of error
ErrorCode	Yes	varchar2(30)	Code error
ErrorLevel	No	number(10)	Error level
ErrorDetail	No	varchar2(2000)	Error detail information

Chapter 6

Membership Schema

This chapter describes the structure of the LDAP Directory Server database used by BillerXpert.

This chapter includes the following sections:

- Membership Schema
- Relationship Between Tables
- Listing of Structures
- Biller Tables
- Customer Tables

CAUTION BillerXpert provides an API that you should use to access the database instead of the schema. The database schema for BillerXpert is subject to change in future versions of BillerXpert. If you rely on the schema, you should consider the potential reimplementation effort that you could incur as the result of an upgrade to the database.

Membership Schema

The database schema for membership consists of tables that represent the following kinds of information:

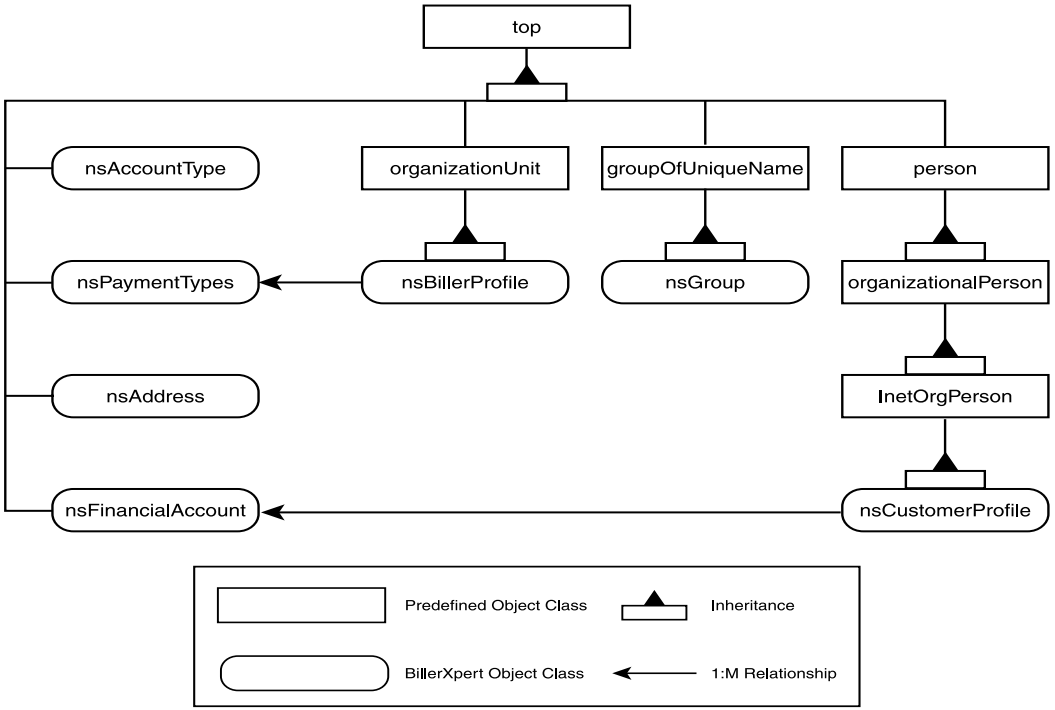
- Biller information
- Financial account information global for the biller site
- Customer information

- Customer financial account information
- Customer payment information

Relationship Between Tables

BillerXpert uses fields from some Directory Server tables, which are also listed in the following membership tables. Figure 6-1 shows the relationships of these tables.

Figure 6-1 Membership Table Relationships



Listing of Structures

Table 6-1 lists the BillerXpert Oracle database tables.

Table 6-1 Membership Tables

Table Name	Functional	Grouping Description
nsAccountType	Biller	Stores financial account information. Global for the biller site.
nsPaymentType	Biller	Stores biller payment information.
nsAddressTable	Customer	Stores addresses of customers.
nsBillerProfile	Biller	Stores biller information.
nsCustomerProfile	Customer	Stores customer information.
nsFinancialAccount	Customer	Stores information about the customer's financial account.
nsGroup	Other	Stores group information for the biller site.

The field type in the membership tables can be one of the following:

- CES (Case Exact String — or case sensitive string)
- CIS (Case Ignore String — or not case sensitive string)
- Telephone

Biller Tables

The Biller tables store information related specifically to the biller or to the biller site.

nsBillerProfile Table

The nsBillerProfile table stores information about the biller. Table 6-2 lists this information.

Table 6-2 nsBillerProfile Table *(1 of 3)*

Name	Required	Type	Description
OU	Y	CIS string	Organization unit

Table 6-2 nsBillerProfile Table *(2 of 3)*

Name	Required	Type	Description
nsName	N	CES string	Biller name.
nsStatus	N	CIS	string Biller status.
nsType	N	CIS string	Biller organization type, for example, Direct.
nsEmail	N	CIS string	Biller email.
nsURL	N	CIS string	Biller URL.
nsChartset	N	CIS string	Character set.
nsLanguage	N	CIS string	Language set
nsCreateddate	N	CIS string	Date the biller was created.
nsCreatedby	N	CIS string	Name of the person who created the biller.
nsCurrency	N	CIS string Currency type.	The default is US dollars.
nsLastModifies	N	CIS string	Name of person who last modified the biller information.
mail	N	CIS string	Biller email address.
nsPayeeID	N	CIS string	Biller payee ID.
nsPrimarycontact	N	CIS string	Biller primary contact name.
nsPrimarycontactPhone	N	CIS string	Biller primary contact phone.
nsPrimarycontactEmail	N	CIS string	Biller primary contact email
nsProfileextension	N	CIS string	Biller profile extension. Used to store additional biller profile information.
nsSecondarycontact	N	CIS string	Biller secondary contact name
nsSecondarycontactPhone	N	CIS string	Biller secondary contact phone
nsSecondarycontactEmail	N	CIS string	Biller secondary contact email
nsBillerCycleDate	N	CIS string	Biller cycle date
nsBillerBillMonth	N	CIS string	Biller bill month
nsCustEnrollOption	N	CIS string	Biller custom enroll option
nsCustBillPresentment	N	CIS string	Biller bill presentment
nsCustAutoPay	N	CIS string	Biller custom auto pay

Table 6-2 nsBillerProfile Table *(3 of 3)*

Name	Required	Type	Description
nsServiceextension	N	CIS string	Biller service extension. Used to store biller defined service options.
nsTimezone	N	CIS string	Biller's time zone
nsAddress1	N	CIS string	Biller's address
nsAddress2	N	CIS string	Biller's address
nsAddress3	N	CIS string	Biller's address 3
nsCity	N	CIS string	Biller's city
nsCountry	N	CIS string	Biller's country
nsTaxid	N	CIS	string Biller's tax identification
nsMerchantid	N	CIS string	Biller's merchant identification
nsgroup	N	CIS string	Biller group
nsInstantGratification	N	CIS string	Biller Instant Gratification
nsInstantGratificationURL	N	CIS string	Biller Instant Gratification URL
nsExtension	N	CIS string	Biller extension

nsAccountType Table

The nsAccountType table stores account type information for a biller. This information is global for the domain.

Table 6-3 lists this information.

Table 6-3 nsAccountType Table *(1 of 2)*

Name	Required	Type	Description
nsGuid	Y	CIS string (primary key)	Global unique ID. Primary key.
cn	N	CIS string	common name
description	N	CIS string	Description
nsOwner	N	CIS string	string Owner
nsServiceOption	N	CIS string	Service option
nsStatus	N	CIS string	Account status

Table 6-3 nsAccountType Table *(2 of 2)*

Name	Required	Type	Description
nsName	N	CES string	Account type name, for example “checking”

nsGroup Table

The nsGroup table stores information about the supported groups. For BillerXpert the supported groups are Administrator and default.

Table 6-4 lists this information.

Table 6-4 nsGroup Table

Name	Required	Type	Description
cn	N	CIS string	common name
description	N	CIS string	Description
nsType	N	CIS string	Type
nsacl	N	CIS string	Access control list
nsstatus	N	CIS string	status

nsPaymentType Table

This table stores payment related information for a customer or biller. Multiple instances of this information can exist for a single customer.

Table 6-5 lists this information.

Table 6-5 nsPayment Table *(1 of 2)*

Name	Required	Type	Description
nsGuid	Y	CIS string (primary key)	Global unique ID. Primary key.
nstype	N	CIS string	Payment type, for example, credit card or check.
nsBrand	N	CIS string	Brand name of the credit card, for example, VISA or Diners.
description	N	CIS string	Description
nsMaxAmount	N	CIS string	Maximum amount

Table 6-5 nsPayment Table *(2 of 2)*

Name	Required	Type	Description
nsminAmount	N	CIS string	Minimum amount
nsName	N	CES string	Payment name, for example, MASTER (short name for a credit card).
nsOwner	N	CIS string	Name of the person responsible for the payment.
nsStatus	N	CIS string	Payment status. These values can be Active or Inactive. Inactive disables the payment method for the customer.

Customer Tables

The customer tables store information on a customer. Some of this information can exist in multiple instances for a single customer. For example, a customer may have more than one financial account from which he pays bills. In this case, the customer will have two or more instances of financial account data.

nsCustomerProfile Table

The nsCustomer table stores information about a customer. Table 6-6 lists this information.

Table 6-6 nsCustomerProfile Table *(1 of 3)*

Name	Required	Type	Description
UID	Y	CIS string	User ID
nsAccount	N	CIS string	Account ID
nsStatus	N	CIS string	Customer status. The values are Active or Inactive.
nsType	N	CIS string	Customer type. This information is biller defined.
nsUrl	N	CIS string	Customer URL.
nsCharset	N	CIS string	Character set.
nsCreateddate	N	CIS string	Date the customer was created.

Table 6-6 nsCustomerProfile Table *(2 of 3)*

Name	Required	Type	Description
nsCreatedby	N	CIS string	Name of person who created the customer.
nsCurrency	N	CIS string	Currency type.
nsLastModified	N	CIS string	Name of person who last modified the customer information.
nsMiddleinitial	N	CIS string	Middle initial of the customer
nsNickName	N	CIS string	Customer nickname.
nsPasswordLastChanged	N	CIS string	Date the password was last changed.
nsPrefix	N	CIS string	Customer prefix, for example, Mr. or Miss.
mail	N	CIS string	Customer email address.
nsPrimarycontact	N	CIS string	Customer primary contact name.
nsProfileextension	N	CIS string	Customer profile extension. The product provides predefined profiles and the biller can define its own.
nsSecondarycontact	N	CIS string	Customer secondary contact name.
nsSecretWord	N	CIS string	Customer secret word. This can be used for additional security purposes. For example, the customer's mother's maiden name.
nsServiceextension	N	CIS string	Customer service extension. These are biller defined.
nsTimezone	N	CIS string	Customer time zone.
nsSuffix	N	CIS string	Customer suffix.
nsAddressID (nsAddress)	N	CIS string	Customer address ID.
nspasswordhint	N	CIS string	Customer password hint.
nsgroup (nsGroup)	N	CIS string	Customer group.
nsAccountType (nsAccountType)	N	CIS string	Customer account type ID.

Table 6-6 nsCustomerProfile Table *(3 of 3)*

Name	Required	Type	Description
nsDefaultPmtacct (nsFinancialAccount)	N	CIS string	Customer default payment account ID.
nsacl	N	CIS string	Access control list.
nsExtension	N	CIS string	Customer extension. Biller defined.
nsFileID	N	CIS string	Customer file ID
nsMaidenName	N	CIS string	Customer's maiden name
nsBirthDate	N	CIS string	Customer's birthdate
nsTaxid	N	CIS string	Customer's tax id
nsdefaultpmtacct	N	CIS string	Customer's default payment account

nsFinancialAccount Table

This table stores information about the customer's financial account. Multiple instances can exist for a single customer.

Table 6-7 lists this information.

Table 6-7 nsFinancialAccount Table *(1 of 2)*

Name	Required	Type	Description
nsGuid	Y	CIS string (primary key)	Global unique ID. Primary key.
description	N	CIS string	Description
cn	N	CIS string	common name
nsPanhandle	N	CIS string	Pan handle.
nsOwner	N	CIS string	Owner
nsName	N	CIS string	Account name
nsPayhandle (nsPaymentType)	N	CIS string	Pay handle
nsStatus	N	CIS string	Account Status
nsType	N	CIS string	Financial account type
nsExtension	N	CIS string	Extension

Table 6-7 nsFinancialAccount Table *(2 of 2)*

Name	Required	Type	Description
nsRoutingNumber	N	CIS string	Routing number
nsAccountNumber	N	CIS string	Account number

nsAddress Table

This table stores the customer address information. Multiple instances can exist for a single customer. Table 6-8 lists this information.

Table 6-8 naAddress Table

Name	Required	Type	Description
cn	N	CIS string	common name
nsGuid	Y	CIS string (primary key)	Global unique ID. Primary key.
nsType	N	CIS string	Address type
nsAddress1	N CIS	string	Address 1
nsAddress2	N	CIS string	Address 2
nsAddress3	N	CIS string	Address 3
nsCity	N	CIS string	City name
nsCountry	N	CIS string	Country name
PostalCode	N	CIS string	Postal code
St	N	CIS string	State
nsStatus	N	CIS string	Status

Appendix A

Enumerated Domains

This appendix describes the domains that an implementation of a component business object interface checks before performing actions against a data repository.

This appendix contains the following sections:

- Domain Checking
- The enumdef.conf File

Domain Checking

BillerXpert has a concept of a domain, which restricts some kinds data in data repositories to specific values. A domain is enforced when access to a data repository occurs, not when the data is set. For example, you can call an IBXBillerProfile object's `setBillerStatus()` method and set any value. If a value outside of the domain is detected while executing `IBXEntity.ejbCreate()` method or the `ejbStore`() method, an exception is raised.

The allowed values are determined by the contents of the enumdef.conf configuration file. For example, the valid values for status are specified by the following lines:

```
"STATUS" NTV {
   "value" StrArr [ "ACTIVE", "INACTIVE", "PENDING", "VOID" ],
   "default" Str "ACTIVE"
```

In this case, the valid values are 'ACTIVE', 'INACTIVE', 'PENDING', and 'VOID'. If a value has not been set, for example, if the setBillerStatus() method was not called before calling the `IBXEntity.ejbCreate()` method, the default value, 'ACTIVE' would be used.

The enumdef.conf File

The enumdef.conf file contains the domains that are used by BillerXpert. Each domain, such as STATUS, has associated with it an array of valid values and a default value.

Table A-1 shows the contents of this file.

Table A-1 enumdef.conf File *(1 of 5)*

```
NTV-ASCII {
    "STATUS" NTV {
    "value"   StrArr [ "ACTIVE", "INACTIVE", "PENDING", "VOID" ],
        "default" Str    "ACTIVE"
},

"CUSTOMERTYPE" NTV {
    "value"   StrArr ["REGULAR", "SPECIAL" ],
        "default" Str    "REGULAR"
},

"DISPUTESTATUS" NTV {
    "value"   StrArr ["NODISPUTE","OPEN", "REOPEN", "CLOSED" ],
        "default" Str   "NODISPUTE"
},

"DISPUTEREASON" NTV {
    "value"   StrArr ["ALREADY_PAID", "INVALID_ITEM",
              "WRONGAMOUNT", "WRONGITEM", "OTHER","" ],
        "default" Str   "ALREADY_PAID"
},

"DOCTYPE" NTV {
    "value"   StrArr ["BILL", "INVOICE", "NOTICE", "STATEMENT" ],
        "default" Str "BILL"
},

"BILLTYPE" NTV {
    "value"   StrArr ["TELCO", "CABLE", "UTILITY", "BANKSTMT",
              "MORTGAGESTMT" ],
        "default" Str   "TELCO"
        },

"COPYTYPE" NTV {
    "value"   StrArr ["ORIGINAL", "DUPLICATE" ],
        "default" Str "ORIGINAL"
},
```

Table A-1 enumdef.conf File *(2 of 5)*

```
"AUTOPAYFREQUENCY" NTV {
    "value"   StrArr ["PAYWHENDUE" ],
        "default" Str "PAYWHENDUE"
},

"PAYMODE" NTV {
    "value"   StrArr ["ONDUEDATE", "BEFOREDUEDATE",
               "ONFIXEDDATE"],
        "default" Str "ONDUEDATE"
},

"PAYAMOUNTTYPE" NTV {
    "value"   StrArr ["FULL", "FIXEDAMOUNT", "MINIMUM",
               "FIXEDPERCENT"],
        "default" Str "FULL"
},

"PENALTYTYPE" NTV {
    "value"   StrArr ["FIXED", "PERCENT" ],
        "default" Str   "FIXED"
},

"PAYSTATUS" NTV {
    "value"   StrArr ["NOT_PAID","FULLY_PAID", "PARTIALLY_PAID",
               "OVER_PAID" ],
        "default" Str "NOT_PAID"
},

"DISCOUNTTYPE" NTV {
    "value"   StrArr ["FIXED", "PERCENT" ],
        "default" Str "FIXED"
},

"MESSAGETYPE" NTV {
    "value"   StrArr ["CORRESPONDENCE", "DISPUTE" ],
        "default" Str "CORRESPONDENCE"
},

"MESSAGESENDERTYPE" NTV {
    "value"   StrArr ["CSR", "ADMIN", "CUSTOMER" ],
        "default" Str "CUSTOMER"
},
```

Table A-1 enumdef.conf File *(3 of 5)*

```
"PMTMODE" NTV {
    "value"   StrArr ["MANUAL","AUTO" ],
        "default" Str   "MANUAL"
},

"BILLERTYPE" NTV {
    "value"   StrArr ["DIRECT","BSP", "CUP" ],
        "default" Str   "DIRECT"
},


"PAYTYPE" NTV {
    "value"   StrArr ["CREDIT CARD","CHECK" ],
        "default" Str   "CREDIT CARD"
},

"CURRENCY" NTV {
    "value"   StrArr ["USD", "CAD", "FRF", "ARS", "ATS", "AUD",
                "BEF", "BRC", "CHF", "CSK", "DEM", "DKK",
                "ESP", "FIM", "GBP", "GRD", "HKD", "INR",
                "IDR", "IEP", "ITL", "JPY", "KRW", "MXP",
                "MYR", "NLG", "NOK", "NZD", "PLZ", "PTE",
                "SEK", "SGD", "ZAR" ],
        "default" Str   "USD"
},

"CARDBRAND" NTV {
    "value"   StrArr ["VISA", "MASTERCARD", "AMEX", "DISCOVER",
                "DINERSCLUB", "CARTEBLANCHE",
                "DDE-CHECKING", "DDE-SAVING"],
        "default" Str   "VISA"
},

"PAYMENTMETHOD" NTV {
    "value"   StrArr ["VISA", "MASTERCARD", "AMEX", "DISCOVER",
                "DINERSCLUB", "CARTEBLANCHE", "CHECK",
                "DDE-CHECKING", "DDE-SAVING"],
        "default" Str   "VISA"
},

DUMMYPAYMENTMETHOD" NTV {
    "value"   StrArr ["DUMMY_SM","DUMMY_CC" ],
        "default" Str   "DUMMY_SM"
},
```

Table A-1 enumdef.conf File *(4 of 5)*

```
""CARDTXNSTATUS" NTV {
    "value"   StrArr ["SUBMITTED", "AUTH_PENDING",
                "AUTH_SUCCESS", "AUTH_FAIL", "DELETED",
                "CAPTURE_PENDING", "CAPTURE_IN_PROCESS",
                "CAPTURE_SUCCESS", "CAPTURE_FAIL",
                "VOID_PENDING", "VOID_SUCCESS", "VOID_FAIL",
                "REFUND_PENDING", "REFUND_FAIL", "NSF" ],
        "default" Str   "SUBMITTED"
},

"CHECKTXNSTATUS" NTV {
    "value"   StrArr ["CHECK_SUB_PENDING", "CHECK_SUB_SUCCESS",
                "CHECK_SUBMITTED", "CHECK_SUB_FAILED",
                "DELETED", "CHECK_SUB_RETRY",
                "CHECK_MAX_RETRY", "CHECK_PENDING",
                "CHECK_SUCCESS", "CHECK_FAILED",
                "CHECK_NO_HOPE"],
        "default" Str   "CHECK_SUB_PENDING"
},

"TXNSTATUS" NTV {
    "value"   StrArr ["SUBMITTED", "CANCELLED", "DELETED",
                "AUTH_PENDING", "AUTH_SUCCESS", "AUTH_FAIL",
                "CAPTURE_PENDING", "CAPTURE_IN_PROCESS",
                "CAPTURE_SUCCESS", "CAPTURE_FAIL",
                "VOID_PENDING", "VOID_SUCCESS", "VOID_FAIL",
                "REFUND_PENDING", "REFUND_FAIL","NSF",
                "CHECK_SUB_PENDING", "CHECK_SUB_SUCCESS",
                "CHECK_SUBMITTED", "CHECK_SUB_RETRY",
                "CHECK_SUB_FAILED", "CHECK_MAX_RETRY",
                "CHECK_PENDING","CHECK_SUCCESS",
                "CHECK_FAILED", "CHECK_NO_HOPE", "FAILED",
                "PENDING"],
        "default" Str   "SUBMITTED"
},

"BATCHTYPE" NTV {
    "value"   StrArr ["GENERAL","SALES","REFUND" ],
        "default" Str   "GENERAL"
},
```

Table A-1 enumdef.conf File *(5 of 5)*

```
"BATCHSTATUS" NTV {
    "value"   StrArr ["PENDING", "SUCCESS", "PARTIAL_SUCCESS",
               "MISMATCH_SUCCESS", "DUPLICATE","RETRY",
               "RESUBMIT","FAILURE", "MAX_SUBMISSIONS",
               "HALTED", "SUSPENDED", "ADMIN_INIT" ],
        "default" Str   "SUCCESS"
},

"TXNTYPE" NTV {
    "value"   StrArr ["DO_AUTH", "DO_VOID", "DO_REFUND",
               "DO_CAPTURE", "CHECK_AUTH"],
        "default" Str   "DO_AUTH"
},

"CHECKUSE" NTV {
    "value"   StrArr ["BUSINESS", "PERSONAL" ],
        "default" Str    "PERSONAL"
},

"PAYMENTVENDOR" NTV {
    "SM"   Str   "SiMulator",
    "CC"   Str   "CyberCash"
},
}
```

Index

A

B

C

E

F

G

I

L

M

N

O

P

R

S

T

U

W