Oracle® Communications Billing and Revenue Management

Calculating Taxes

Release 7.5

E16722-08

August 2016

ORACLE®

Oracle Communications Billing and Revenue Management Calculating Taxes, Release 7.5

E16722-08

Contents

2 Setting Up Pipeline Manager Taxation

3 Setting Up Tax Calculation for Vertex

4 Customizing Tax Calculation

5 Calculating Taxes Utilities

Preface

This book describes how to configure the taxation software in Oracle Communications Billing and Revenue Management (BRM).

Audience

This document is intended for those tasked with configuring BRM taxation software and enabling it to work with third-party products.

Downloading Oracle Communications Documentation

Product documentation is located on Oracle Help Center:

http://docs.oracle.com

Additional Oracle Communications documentation is available from the Oracle software delivery Web site:

https://edelivery.oracle.com

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Document Revision History

The following table lists the revision history for this book.

Version	Date	Description
E16722-01	November 2011	Initial release.
E16722-02	May 2012	Minor formatting and text changes.

Version	Date	Description
E16722-03	August 2013	On HP-UX IA64, BRM 7.5 is certified as of BRM 7.5 Patch Set 5. Documentation added for HP-UX IA64.
E16722-04	February 2014	Documentation updates for BRM 7.5 Patch Set 7. ▪ Minor formatting and text changes.
E16722-05	August 2015	Documentation updates for BRM 7.5 Maintenance Patch Set 1. ▪ Removed Taxware information in the following chapters: About Calculating Taxes Customizing Tax Calculation ▪ Removed chapter, Setting Up Tax Calculation for Taxware. ▪ Updated the "Providing Entries for Vertex Tax Codes in the taxcodes_map File" section.
E16722-06	December 2015	Documentation updates for BRM 7.5 Patch Set 14. ▪ Updated the "Tax Information Stored for Each Event" section.
E16722-07	April 2016	Documentation updates for BRM 7.5 Patch Set 15. ▪ Added the following section: Numbering Differences in Vertex-to-BRM Jurisdiction Code Mapping
E16722-08	August 2016	Documentation updates for BRM 7.5 Patch Set 16. ▪ Updated the following sections: Configuring the Vertex DM for Communications Tax Q Series Source Code for Customizing PCM_OP_RATE_POL_PRE_TAX

1

About Calculating Taxes

This chapter describes how Oracle Communications Billing and Revenue Management (BRM) calculates taxes.

Before reading this document, you should be familiar with BRM architecture, real-time rating, and Pipeline Manager. For information, see the following documents:

- "BRM System Architecture" in *BRM Concepts*
- "About Real-time Rate Plans" in *BRM Setting Up Pricing and Rating*
- "About Pipeline Rating" in *BRM Configuring Pipeline Rating and Discounting*

About Taxation

BRM can calculate taxes for events that are rated in real time and by pipeline batch rating. You can choose whether taxes are calculated using third-party tax software or within BRM. You can also choose to apply taxes during real-time rating, during pipeline batch rating, during billing, or a combination of the three.

Important: BRM applies taxes to currency resources only. It does not apply taxes to non-currency resources.

Choosing the Tax Calculation Method

You can configure the method that BRM uses to calculate taxes. There are three possibilities:

- **Third-party tax software**. You use third-party tax software to apply complex taxation for multiple jurisdictions. The software keeps track of changing tax codes and jurisdictions for you.

  See "Calculating Taxes by Using Third-Party Tax Software" for more information.

  Note: This is the default tax calculation method.

- **BRM policy opcodes**. You use BRM taxation policy opcodes to apply custom taxation that does not require the full capabilities of third-party tax software.

  See "Calculating Taxes by Using Policy Opcodes" for more information.

- **BRM tax codes file**. You use the tax codes file to apply simple flat taxes.

  See "Calculating Flat Taxes by Using the taxcodes_map File" for more information.

Choosing When to Calculate Taxes

You can configure BRM to calculate taxes at one or more of the following times:

- **During real-time rating**. In this configuration, taxes are calculated when the event occurs, and they are added to the customer's account balance in real time. This way, you always have an accurate reading of a customer's account balance at any time in the accounting cycle.

  See "About Specifying When to Tax Events" for more information.

- **During pipeline batch rating**. In this configuration, taxes are calculated for events as they are rated in a pipeline. Taxes are added to account balances at the same time as other balance impacts from pipeline batch rating. Taxing events during pipeline rating allows you to generate pipeline output files that include both usage and tax charges.

  See "About Pipeline Taxation" for more information.

- **During billing**. Deferring tax calculation to the billing process reduces rounding errors because all events of the same type are calculated together. For example, taxes are calculated on the total amount of usage fees rather than on individual usage events.

  See "About Specifying When to Tax Events" for more information.

About Pipeline Taxation

BRM can apply taxes to events rated by Pipeline Manager. You can configure your system to tax pipeline events in one of the following ways:

- **During the pipeline rating process**. Taxing events during pipeline rating allows you to generate pipeline output files that include both usage and tax charges. This is useful when output files are not sent through the billing process. For example, when you rate outcollect roaming calls, you rate calls from another carrier's customer and then send the pipeline output files directly to the other carrier for rating and verification.

  Important: You can apply only flat taxes during pipeline rating. For complex taxation, you must defer taxation to the billing process.

- **During the billing process**. Deferring taxation to the BRM billing process allows you to apply complex tax rules by using third-party tax software or customized BRM policy opcodes. It also reduces rounding errors because BRM taxes all events of the same type at the same time. This configuration works best for most wireless applications because it allows complex taxation.

- **During both pipeline rating and billing**. Taxing events during both pipeline rating and BRM billing allows you to:

  – Obtain itemized tax charges for each usage event.

  – Tax non-usage events, such as one-time purchase and cycle-time events, that are rated in real time rather than by Pipeline Manager.

See "Setting Up Pipeline Manager Taxation" for more information.

About Tax Calculation for Account Groups

In an account group, parent accounts always pay taxes for accounts with nonpaying (subordinate) bill units. However, there are two ways to calculate taxes. Taxes can be calculated for each individual nonpaying child bill unit and can be listed as separate items on the parent bill, or taxes can be consolidated into a single item for both the parent and subordinate bill units.

You specify how BRM calculates taxes for account groups by entering one of the following values in the **cycle_tax_interval** entry in the Connection Manager (CM) configuration (**pin.conf**) file:

- **accounting** (default): BRM calculates taxes separately for the parent and each subordinate bill unit and then lists the taxes as separate items on the parent bill.
- **billing**: BRM rolls activities for each subordinate bill unit into the parent account and calculates taxes for the parent account only. The single tax item on the parent account includes taxes from both the parent and subordinate bill units. This option improves overall performance of BRM tax computation.

Important: Bill Now, regardless of the option you select for the entry, always calculates taxes this way.

Note: When Bill Now is run on the child account, the PIN_FLD_GROUP_OBJ field in the **/event/billing/cycle/tax** object contains the POID of the subordinate bill unit. When Bill Now is run on the parent account, the field contains the POID of the parent account's bill unit. This field is not populated during regular billing.

About Tax Calculation for Adjustments, Disputes, and Settlements

You can specify whether to perform an adjustment, dispute, or settlement with or without taxes, provided the original item or event was taxable. Taxed adjustments, disputes, and settlements effectively reverse whatever tax was levied on the original item. See "Configuring Adjustments, Disputes, and Settlements" in *BRM Managing Accounts Receivable* for more information.

Calculating Taxes by Using Third-Party Tax Software

When using third-party tax software, BRM calculates taxes as follows:

1. BRM rates a billable event and then collects information about the event that can be used for calculating taxes, such as the location of your business, the location of the customer, and customer tax exemptions.

   See "About Supplying the Data Needed for Calculating Taxes" for more information.

2. A BRM tax manager compiles the tax calculation data into a transaction and sends it to your third-party tax software.

   See "About BRM Tax Managers" for more information.

3. The third-party tax software calculates the taxes on the event and returns the tax amount to BRM.

   See "About Third-Party Tax Software" for more information.

4. BRM adds the tax amount to the customer's account balance.
5. BRM records the tax amount as general ledger (G/L) data.

   See "Planning Your G/L IDs" in *BRM Collecting General Ledger Data* for more information.

About Third-Party Tax Software

You can use one or more of the following third-party tax software packages with BRM:

- **Vertex Sales Tax Q Series**: This software calculates sales and use taxes for the United States and Canada.
- **Vertex Communications Tax Q Series**: This software calculates telecommunications taxes for the United States and Canada.

> **Important:** Third-party tax software is *not* included with BRM. You must separately obtain and install the software you want.

For detailed information about each package, see its documentation.

About BRM Tax Managers

BRM uses tax managers to communicate with third-party tax software. The tax manager compiles all tax-related data for an event and then sends the data to your third-party tax software.

Table 1–1 lists the supported third-party tax software packages and the corresponding BRM tax managers. BRM tax managers are optional features that you install separately.

Table 1–1 Supported Third-Party Tax Software Packages

Tax Software	BRM Tax Manager
Vertex Communications Tax Q Series	Vertex Manager. To configure Vertex Manager, see "Setting Up Tax Calculation for Vertex" for details.
Vertex Sales Tax Q Series	Vertex Quantum Manager or Vertex Manager. To configure Vertex Manager or Vertex Quantum Manager, see "Setting Up Tax Calculation for Vertex" for details.

> **Note:** BRM has its own auditing features. It does not support the auditing features of Vertex Communications Tax Q Series, or Vertex Sales Tax Q Series. The auditing features in these tax packages do not handle all of the information that is stored in BRM.

About Supplying the Data Needed for Calculating Taxes

Some of the data that the tax software uses for calculating taxes is taken from the event itself. For example:

- The amount charged or credited.
- The currency used when charging for the event.

- The date of the event.
- The location of the customer (for tax purposes, this is called the *shipped-to location*).

This information is obtained from the event and from the account; you do not need to configure anything to include it in the data sent to the tax software. However, you do need to configure BRM to supply the following data:

- Tax codes. See "About Specifying Tax Codes" for more information.
- Tax suppliers. See "About Tax Suppliers" for more information.
- Tax exemptions. See "About Tax Exemptions" for more information.

About Specifying Tax Codes

A tax code indicates which tax to apply based on the BRM product. For example, a t-shirt uses a different tax code than an online service subscription. Each tax package includes its own tax codes, called *product codes*. Instead of using these product codes, which are numeric and therefore not descriptive, you can create and use your own tax codes and map them to the tax software's product codes.

To implement tax codes, you define them in the **taxcodes_map** file (*BRM_Home*/**sys/cm** is the default location). By default, the **taxcodes_map** file supplies BRM with the following information about your third-party software.

- **Pkg**: The tax package you use:
  - C – Vertex Communications Tax Q Series (telecommunications).
  - Q – Vertex Sales Tax Q Series (sales and use).
  - U – User-defined for a custom tax calculation. See "Calculating Taxes by Using Policy Opcodes" or "Calculating Flat Taxes by Using the taxcodes_map File" for more information.
- **Code 1**: The field value is determined by the tax package you use.
  - For Vertex Communications Tax Q Series, this field is the category code. See the Communications Tax Q Series documentation for more information about its category codes.
  - For Vertex Sales Tax Q Series, this field is the transaction type. See the Sales Tax Q Series documentation for more information on transaction types.
- **Code 2**: The field value is determined by the tax package you use.
  - For Vertex Communications Tax Q Series, this field is the service code. Refer to the Communications Tax Q Series documentation for more information on service codes.
  - For Vertex Sales Tax Q Series, this field is the transaction subtype. Refer to the Sales Tax Q Series documentation for more information on transaction subtypes.
- **Si** (sales indicator):
  - For Vertex Communications Tax Q Series and Vertex Sales Tax Q Series, this is the resale flag field, which indicates if the product is for sale (**S**) or resale (**R**). The default is **S**.

See your tax package documentation for information about product codes, service codes, and service indicators.

The following is an example of a **taxcodes_map** file:

```
#   Taxcode           Pkg   Code1   Code2   Si   Wt_code   Cmdty_code
#   ---------------   ---   -----   -----   --   -------   ----------
#   usage2          :  Q  :    01 :    01 :  S  :        :
#   direct          :  C  :    01 :    01 :  S  :        :
#   toll            :  C  :    04 :    02 :  S  :        :
#   direct2         :  B  :    01 :    04 :  S  :        :
#   toll2           :  B  :    01 :    01 :  S  :        :
#   user_code       :  U  :  10.0
monthlyCard         :  Q  :    03 :    03 :  S  :        :
installVertex       :  Q  :    03 :    03 :  S  :        :
interCommtax        :  C  :    04 :    01 :  S  :        :
chet_is_cool        :  C  :    04 :    01 :  S  :        :
intraCommtax        :  C  :    01 :    02 :  S  :        :
ipUsage             :  Q  :    03 :    03 :  S  :        :
```

About Tax Suppliers

A tax supplier is the company or corporate division responsible for collecting taxes for a given transaction. Tax suppliers typically include your corporate headquarters and branch offices. Third-party tax software uses the tax supplier to calculate taxes.

Tax supplier data can include the following information:

- The name of the tax supplier.
- The address to use as the ship-from locale. See "About Tax Suppliers and Locales" for more information.
- Whether the tax supplier is the default tax supplier.
- The tax nexus state. You can have a tax nexus in any state where your company has a substantial presence, as defined by federal tax laws.
- The VAT certificate number.

You specify how to supply tax supplier data to your tax software in one of the following ways:

- Provide the tax supplier data with each transaction that you send to the tax software. For example, each transaction might include the ship-from locale based on the tax supplier address and the applicable VAT certificate. See "Providing the Tax Supplier Data with Each Transaction" for more information.
- Allow the tax software to determine the correct tax supplier based on the transaction data plus the company and business locations you give the tax software. You provide information to the tax software by running the tax software's toolkits. The tax software then determines which ship-from locale to use, if there is a tax nexus presence, and which VAT certificate to use. See "Allowing Your Sales and Use Tax Software to Determine Tax Supplier Data" for more information.

About Tax Suppliers and Locales

Tax software typically calculates taxes by using four locales:

- **Shipped-to**: The address of the customer who made the purchase. BRM obtains the shipped-to address from the customer's account. For telecommunications taxation, this is the termination number.
- **Shipped-from**: The address of the company supplying the product. For telecommunications taxation, this is the origination number.

- **Bill origin**: The location from which the product was shipped (also called *point of origin*); for example, a warehouse. This is not used for telecommunications taxation.
- **Bill approval**: The location where the order for the product was taken (also called *point of acceptance*); for example, the location of a customer service representative who registers your customers. For telecommunications taxation, this is the charge-to number.

For sales and use taxes, BRM does not differentiate between ship from, bill origin, and bill approval. Therefore, when providing the tax software with data, BRM uses the same value for all three locales. This value is derived from the tax supplier's address. For telecommunications taxation, the numbers are derived from call details records (CDRs).

> **Note:** For telecommunications taxation, the shipped-to, bill origin, and bill approval locales are not used and are therefore stored in BRM with a NULL value.

For information on customizing the tax locales, see "Retrieving Tax Calculation Data" for more information.

Providing the Tax Supplier Data with Each Transaction

If your business uses only one tax supplier, you can define a default tax supplier to use for all products. You do not need to enter any tax supplier information when creating products in Pricing Center.

You can specify multiple tax suppliers. Use Pricing Center to assign each product in your price list to a tax supplier. When an event is rated, the tax supplier defined in the product is sent to the tax software.

For example, if you offer different products in different states, you can indicate the state in which the product was sold by choosing the tax supplier for that state.

Allowing Your Sales and Use Tax Software to Determine Tax Supplier Data

If your business has customers in multiple countries or operates under other conditions where complex tax laws are involved, you should allow your sales and use tax software to choose the correct tax supplier. The disadvantage is that you need to include every BRM product in the **tax_supplier_map** file, and you need to update the file when you add or delete products.

The **tax_supplier_map** file allows your tax package to determine the correct tax supplier company ID and business location, based on the product and the ship-to locale.

The ship-from address that you define in the **tax_supplier_map** file can be different from the address used when you defined the tax supplier. In this case, your tax package obtains the address from the **tax_supplier_map** file. This allows you to create multiple ship-from addresses for a single tax supplier.

For Sales Tax Q Series, the **tax_supplier_map** file also maps BRM tax supplier names to your company IDs stored in the company profile, Sales Tax Q Series TDM. BRM uses this information to maintain database integrity.

For telecommunications tax only, the **Reg** (regulated) column identifies the billing company as regulated (**0**) or unregulated (**1**). This flag determines whether the taxes of unregulated or regulated utilities are calculated.

The following example shows part of a **tax_supplier_map** file:

```
Product    ShipTo     Company ID     Business loc  Ship From                Reg
_____________________________________________________________________________
Item_One : ;;CA;US : Tax supplier1 : loc_1       : Cupertino;CA;95014;US :  1
Sub_One  : ;;;US   : Tax supplier1 : loc_1       : Cupertino;CA;95014;US :  1
Sub_One  : ;;;FR   : Tax supplier2 : loc_2       : Paris;;;FR            :  0
```

About Tax Exemptions

When sending data to the tax software, BRM includes tax exemptions for the customer.

Use Customer Center to specify tax exemptions for each account. For example, you can specify if an account is exempt from city taxes. You can also specify the percentage of the exemption. For example, if 10% of the amount is not taxed, enter **10**. For a total exemption, type **100**.

See Customer Center Help for more information on adding, changing, and deleting tax exemption information.

Note:

- You can supply tax exemptions only after the account is created. Therefore, tax exemptions do not apply to charges incurred when you create the account.
- Communications Tax Q Series does not support partial tax exemptions in percentage. It supports only full exemption (100%) or no exemption (0%). If you set up an account to have partial tax exemption, BRM sends the transaction for this account to the tax package as 100% tax exempt.

Updating Your Tax Software

If you use a tax DM to obtain tax data, you should update the configuration files periodically. These files, which you obtain from the company that makes your tax software, contain updated tax information such as tax tables that have changed because of new state or national laws. Check with your tax software representative for information about obtaining updates.

Calculating Taxes by Using Policy Opcodes

If your business does not require complex tax calculation, you can use the built-in capabilities of BRM to define tax rates. You can also modify those capabilities for more complex situations. See "Using Custom Tax Rates" for more information.

Calculating Flat Taxes by Using the taxcodes_map File

You implement simple flat taxes directly in the **taxcodes_map** file. You enter one or more tax rates for each tax code. You can use several criteria, such as validity dates and jurisdictions, to specify more than one rate for each tax code.

Note: To define flat taxes, you use a different format in the **taxcodes_map** file than you use for third-party tax calculation software.

Important: Each tax code entry must have **U** in the **Pkg** column of the **taxcodes_map** file. The **U** entry causes the PCM_OP_CUST_POL_TAX_INIT policy opcode to cache the tax rate information. These rates are applied by the PCM_OP_CUST_POL_TAX_CALC policy opcode.

In the following sample **taxcodes_map** file, the highlighted entry is for a 4.5% VAT that is valid from February 1, 2002, to January 31, 2008, in Great Britain.

```
# Taxcode     Pkg  Rate   Start      End        Lvl   List         Descr      Rule
# -------     ---  -----  -----      ---        ---   ----         -----      ----
  usage     : U :  4.0  : 02/01/01 : 01/31/02 : Fed : US         : USF      : Std
  cycle     : U :  3.5  : 02/01/02 : 01/31/03 : Fed : US         : Excise   : Std
  purchase  : U :  8.25 : 02/01/02 : 01/31/03 : Sta : CA         : Sales    : Std
  toll      : U :  2.0  : 02/01/02 : 01/31/03 : Fed : US         : TRS      : Tax
  toll      : U :  1.5  : 02/01/02 : 01/31/03 : Sta : CA         : 911      : Tax
  toll      : U :  2.0  : 01/01/02 : 01/01/10 : Sta : CA         : B&O      : Tax
  toll      : U :  3.15 : 01/01/00 : 01/01/10 : Cit : Cupertino  : Deaf     : Tax
  VAT       : U :  5.0  : 02/01/01 : 01/31/02 : Fed : GB;FR      : VAT-EU   : Std
  VAT       : U :  4.5  : 02/01/02 : 01/31/08 : Fed : GB         : VAT-GB   : Std
  VAT       : U :  4.0  : 02/01/02 : 01/31/08 : Fed : FR         : VAT-FR   : Std
```

Table 1–2 provides descriptions of what you can enter in the each of the columns in the **taxcodes_map** file:

Table 1–2 taxcodes_map Columns

Column	Explanation
Taxcode	The tax code. A unique alphanumeric value that defines categories with different tax treatments.
Pkg	Package code. For custom tax policies, this is always **U**, for user-defined.
Rate	The tax rate in percent. **4.25** means 4.25%, for example. For prepaid purchase events that grant negative currency resources, the corresponding tax associated with it should also be negative. See the following example: `PREPAID_PURCH_TAX : U : -4.5 : 02/01/02 : 01/31/08 : Fed : GB : VAT-GB : Std`
Start	Start date of the validity period for the tax rate in *mm/dd/yy* format.
End	End date of the validity period for the tax rate in *mm/dd/yy* format.
Lvl	Jurisdiction level for which this rate is applicable. Values are: **Fed -** Federal level **Sta -** State level **Cou -** County level **Cit -** City level
List	List of jurisdiction values applicable for this rate. Similar to a nexus for the corresponding jurisdiction level. For example, if the **Lvl** value is **Sta**, the **List** values must be state-level jurisdictions. Separate list values by a semicolon (;). To make the value applicable to all, use an asterisk (*).
Descr	Text description of the tax.

Table 1–2 (Cont.) taxcodes_map Columns

Column	Explanation
Rule	Determines how taxes will be computed. Values are: **Std** - Standard tax computation. Taxes are computed based on the taxable amount and are then added to the total. **Tax** - "Tax on tax" computation. Taxes are computed based on previous taxable amounts and taxes and are then added to the total. For example, if tax1 = 10%, tax2 = 20%, and charge = 100.00, taxes are computed as follows: tax1 = 10% @ 100.00 = 10.00 tax2 = 20% @ (100.00 + 10.00) = 22.00 **NCS** - Non-cumulative standard tax computation. Taxes are computed based on the taxable amount but are not added to the total. **NCT** - Non-cumulative "tax on tax" computation. Taxes are computed based on the taxable amount but are not added to the total. For example, if tax1 = 10%, tax2 = 20%, and charge = 100.00, taxes are computed as follows: tax1 = 10% @ 100.00 = 10.00 tax2 = 20% @ 100.00 = 20.00

To define tax rates:

1. Ensure that your CM configuration file (**pin.conf**) contains an entry specifying the location of the **taxcodes_map** file. For example:

   ```
   - fm_rate taxcodes_map ./taxcodes_map
   ```

2. Open the **taxcodes_map** file, which is located by default in the *BRM_Home***/sys/cm** directory.
3. Edit the **taxcodes_map** file to contain the tax codes and rates used for your products.
4. Save and close the file.
5. Stop and restart the CM.

   See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide* for more information.

About Defining Tax Suppliers

Tax suppliers are saved in BRM as **/config/tax_supplier** objects. You must define at least one tax supplier for your system, even if you use the BRM database as the primary source of tax supplier information. If you define multiple tax suppliers, you can have only one default tax supplier.

If you use a third-party tax calculation software, the procedure is slightly different depending on the software. See "Setting up Tax Supplier Information" for information on Vertex.

Defining Tax Suppliers and Loading Them into the BRM Database

You define tax suppliers by editing the *BRM_Home***/sys/data/config/pin_tax_supplier.xml** file. You load tax suppliers into the database by running the "load_tax_supplier" utility.

You cannot add or change tax suppliers individually. Each time you run **load_tax_supplier**, you overwrite existing data with the entire contents of the **pin_tax_supplier.xml** file. To modify or remove existing tax suppliers, edit or remove their information in the file before loading it.

To define and load tax suppliers:

1. Open the **pin_tax_supplier.xml** file in an XML editor or a text editor.

   By default, this file is located in *BRM_Home***/sys/data/config**.

2. Add a **TaxSupplierElement** child element to the **TaxSupplierConfiguration** parent element for each tax supplier.

   A **TaxSupplierElement** looks like this:

```
<TaxSupplierElement>
   <Name>supplier_name</Name>
   <Description>supplier_description</Description>
   <Address>supplier_address</Address>
   <NexusInfo>nexus_values</NexusInfo>
   <RegulatedFlag>flag_value</RegulatedFlag>
   <DefaultFlag>flag_value</DefaultFlag>
   <VATInfo>
      <CanonCountry>country_code</CanonCountry>
      <VATCertificate>certificate_number</VATCertificate>
   </VATInfo>
</TaxSupplierElement>
```

3. Set values for the tags in the **TaxSupplierElement** child element.

   See "Entries in the pin_tax_supplier.xml File" for information about tag values.

4. Save and close the file.

5. Enter the following command to load the tax supplier data:

```
load_tax_supplier pin_tax_supplier.xml
```

   Important: If you are not working in the same directory as the **pin_tax_supplier.xml** file, include the complete path to the file. For example:

```
load_tax_supplier BRM_Home/sys/data/config/pin_tax_supplier.xml
```

6. Stop and restart the CM.

   See "Starting and Stopping the BRM System" in *BRM System Administrator's Guide* for more information.

To verify that the **pin_tax_supplier.xml** file was loaded, you can display the **/config/tax_supplier** object by using the Object Browser, or use the **robj** command with the **testnap** utility. (See "Reading an Object and Writing its Contents to a File" in *BRM Developer's Guide*.)

Entries in the pin_tax_supplier.xml File

By default, the **pin_tax_supplier.xml** file is located in *BRM_Home***/sys/data/config**. Table 1–3 describes the **pin_tax_supplier.xml** file.

Table 1–3 pin_tax_supplier.xml File

Entry	Description
Name	Enter the tax supplier name. This name is the company ID in the **tax_supplier_map** file. This name appears in Pricing Center as the tax supplier ID. **Vertex** - This name is the company ID in the TDM or company file. If you use the TDM, the entry will be a four-character code. The name must match exactly.
Description	Enter a description for the tax supplier.
Address	Enter the address of the tax supplier. Use the following format: *city*;*state*;*zip_code*;*country*
NexusInfo	Enter a semicolon-delimited list of states in which the tax supplier has a tax nexus. If you use third-party taxation software, enter an asterisk (*) to allow Vertex to determine the tax nexus. See "Setting Up Tax Calculation for Vertex" for more information.
RegulatedFlag	**Vertex**: For Communications Tax Q Series, enter whether the utility is regulated (**1**) or unregulated (**0**).
DefaultFlag	Enter **1** for default the tax supplier; enter **0** for all other tax suppliers. If you have multiple tax suppliers, only one can be the default.
CanonCountry	Enter the country name. See the Vertex documentation for a list of valid values.
VATCertificate	Enter the VAT certificate number associated with the tax supplier. BRM currently does not use this entry. This entry can be used in a custom implementation of tax calculation.

Example pin_tax_supplier.xml File

The following **pin_tax_supplier.xml** file contains two tax suppliers; the first is the default tax supplier.

```
<TaxSupplierConfiguration>
  <TaxSupplierElement>
     <Name>TS_1</Name>
     <Description>Tax Supplier 1</Description>
     <Address>Cupertino;CA;95014;US</Address>
     <NexusInfo>*</NexusInfo>
     <RegulatedFlag>0</RegulatedFlag>
     <DefaultFlag>1</DefaultFlag>
     <VATInfo>
        <CanonCountry>US</CanonCountry>
        <VATCertificate>vat_cert_US</VATCertificate>
     </VATInfo>
     <VATInfo>
        <CanonCountry>UK</CanonCountry>
        <VATCertificate>vat_cert_UK</VATCertificate>
     </VATInfo>
  </TaxSupplierElement>
  <TaxSupplierElement>
     <Name>TS_2</Name>
     <Description>Tax Supplier 2</Description>
     <Address>New York;NY;10013;US</Address>
```

```
      <NexusInfo>*</NexusInfo>
      <RegulatedFlag>0</RegulatedFlag>
      <DefaultFlag>0</DefaultFlag>
      <VATInfo>
         <CanonCountry>US</CanonCountry>
         <VATCertificate>vat_cert_US</VATCertificate>
      </VATInfo>
      <VATInfo>
         <CanonCountry>UK</CanonCountry>
         <VATCertificate>vat_cert_UK</VATCertificate>
      </VATInfo>
   </TaxSupplierElement>
</TaxSupplierConfiguration>
```

About Specifying When to Tax Events

BRM can apply taxes during one or more of these processes:

- Real-time rating
- Billing
- Pipeline rating

To enable taxation during real-time rating or billing, see "Enabling Taxation During Real-Time Rating or Billing" for more information.

You specify whether to tax events during the pipeline rating process by configuring the pipeline to include particular modules. See "Setting Up Pipelines to Tax Events" for more information.

Enabling Taxation During Real-Time Rating or Billing

You specify whether to tax events during the real-time rating process or the billing process in two ways:

- **Globally**: You enable BRM to perform real-time and/or billing (cycle-time) taxation by using the Connection Manager (CM) configuration file.

  See "Enabling Taxation Globally" for more information.

- **Per Event Type**: You can specify when specific events are taxed by using your real-time rate plans.

  See "Specifying Taxation In Your Real-Time Rate Plans" for more information.

Enabling Taxation Globally

You can turn BRM taxation on and off globally by using the **taxation_switch** entry in the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**). You can choose to enable real-time tax calculation, deferred tax calculation, or both (the default setting). You can also disable tax calculation entirely.

> **Note:** Because deferred taxation procedures are triggered even if BRM is not configured to use taxation features, turning taxation off can improve performance. If BRM is configured for taxation, you can turn taxation off and on for testing.

Important: When real-time discounting is enabled, you must also set the FCT_Discount module registry entry **TaxationMode** to be the same as the **taxation_switch** entry so that tax is calculated based on the amount due after discounts have been applied. Otherwise, tax is calculated based on the original due amount.

To enable or disable taxation globally:

1. Open the Connection Manager (CM) configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. Set the value of the **taxation_switch** entry.

   You can use any of the following values:

   0 - Tax calculation is entirely disabled.

   1 - Only real-time tax calculation is enabled.

   2 - Only deferred (cycle-time) tax calculation is enabled.

   3 - (Default) Both real-time and deferred tax calculation are enabled.

   For example:

   ```
   - fm_bill taxation_switch 1
   ```

3. Save and close the file.
4. Stop and restart the CM.

   See the discussion of starting and stopping BRM in *BRM System Administrator's Guide* for more information.

Specifying Taxation In Your Real-Time Rate Plans

You can specify the following for each rate plan that you create in Pricing Center:

- The tax code to apply.
- When to apply the tax code: during real-time rating or billing.

To assign taxation to specific rate plans:

1. Open a real-time rate plan in Pricing Center.
2. In the Rate Plan Properties dialog box, specify when to calculate taxes and which tax code to apply as shown in Figure 1–1.

Figure 1–1 Rate Plan Properties Dialog Box

For more information on how to specify taxation in a rate plan, see Pricing Center Help.

Recording Taxes in the General Ledger (G/L)

You can record taxes as general ledger data. To do so:

- Create a general ledger ID (G/L ID) for each tax code.
- Run the General Ledger report with the Tax attribute.

See "About Collecting General Ledger Data" in *BRM Collecting General Ledger Data* for more information.

Reversing Tax Balance Impacts

When you make an adjustment in the Event Browser, the adjustment can include taxes. See Event Browser Help for more information.

Tax Information Stored for Each Event

For each taxable event, BRM can store the following information:

- The currency resource used to calculate the tax. This is always the account currency.
- The tax code of the rate used to charge for the event.
- The amount charged for the event before taxes are applied.
- The G/L ID associated with the tax code. See "Planning Your G/L IDs" in *BRM Collecting General Ledger Data* for details.
- The ship-to address (location where the tax is charged).
- The ship-from address (location where the tax originated).
- Bill-acceptance address.
- Bill-origin address.
- The tax exemption type.
- The name of the tax type, such as state or local.
- The amount of taxation.
- The tax rate, stored as a percentage.
- Whether the tax is sales, use, or rental.

Note: For zero balance impact events or free events, BRM stores only the address information.

Configuring How BRM Calculates Taxes

You can configure several aspects of BRM tax treatment. These include:

- Validating customer addresses to ensure that BRM uses the correct ZIP code for tax calculation.

  See "About Validating Customer Addresses" for more information.

- Reporting zero tax amounts for jurisdictions that require this type of reporting.

  See "Reporting Zero Tax Amounts" for more information.

- Itemizing taxes by jurisdiction.

  See "Itemizing Taxes by Jurisdiction" for more information.

Important: You can also configure how BRM treats taxes for adjustments, disputes, and settlements. See "Configuring Taxes for Adjustments, Disputes, and Settlements" in *BRM Managing Accounts Receivable* for more information.

About Validating Customer Addresses

When taxes are calculated for customers in the United States, the tax rate is partially determined by the customer's ZIP code. If the customer's ZIP code is not correct and fails validation, or it is for a different state, the tax software calculates the taxes as **0**.

To avoid this problem, you can configure BRM to validate the customer's state and ZIP code when the customer registers.

Some third-party tax packages provide address validation. You can enable address validation by specifying which package to use for the **tax_valid** entry in the CM **pin.conf** file. The validation performed depends on the software you specify:

- Vertex Sales Tax Q Series validates city, state, and ZIP code.
- Vertex Communications Tax Q Series does not support address validation. If you configure BRM to use Communications Tax Q Series for validation, it always returns a positive result.

To specify a tax package to validate addresses:

1. Open the Connection Manager (CM) configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. Set the value of the **tax_valid** entry.

   You can use any one of the following values in Table 1–4:

Table 1–4 tax_valid Values

Value	Description
0	BRM does not check the ZIP code. Default value.
3	BRM uses Vertex Sales Tax Q Series to validate the ZIP code. (The Vertex DM must be running.)
4	No effect. BRM uses Communications Tax Q Series to validate the ZIP code, but Communications Tax Q Series does not perform any validation. A positive result is always returned.

   For example:

```
- fm_cust_pol tax_valid 3
```

3. Save and close the file.
4. Stop and restart the CM.

   See the discussion of starting and stopping BRM in *BRM System Administrator's Guide*.

Reporting Zero Tax Amounts

By default, when a zero amount is returned from taxation, BRM does *not* report the tax amount and the TAX_JURISDICTIONS array is not created for the event. However, legal requirements in some jurisdictions demand that zero tax amounts be reported.

To configure BRM to report zero taxes for jurisdictions:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. Uncomment the **fm_rate include_zero_tax** entry and change the value of the entry to **1**:

   ```
   - fm_rate include_zero_tax 1
   ```

3. Save and close the file.
4. Stop and restart the CM.

   See the discussion of starting and stopping BRM in *BRM System Administrator's Guide*.

If multiple taxes, including zero taxes, are reported from the same jurisdiction, they are aggregated by default. This may hide the effect of reporting zero taxes. To prevent zero taxes from being hidden in this way, you should configure BRM to itemize taxes by jurisdiction. See "Itemizing Taxes by Jurisdiction" for more information.

Itemizing Taxes by Jurisdiction

To configure BRM to itemize all taxes by jurisdiction:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. Change the value of the following entry to **itemize**:

   ```
   - fm_rate tax_return_juris itemize
   ```

3. Save and close the file.
4. Stop and restart the CM.

   See the discussion of starting and stopping BRM in *BRM System Administrator's Guide*.

2

Setting Up Pipeline Manager Taxation

This chapter describes how to configure your Oracle Communications Billing and Revenue Management (BRM) system to tax events rated by a batch pipeline.

Before reading this document, you should be familiar with BRM architecture, BRM taxation, and pipeline batch rating. For information, see the following documents:

- "BRM System Architecture" in *BRM Concepts*
- About Calculating Taxes
- "About Pipeline Rating" in *BRM Configuring Pipeline Rating and Discounting*

Configuring Your System to Tax Events Rated by a Batch Pipeline

To configure your system to tax events rated by a batch pipeline, perform the following steps:

1. Enable taxation in your Pipeline Manager rate plans.

   See "Enabling Taxation in Pipeline Rate Plans" for more information.

2. Set up your tax rates.

   See "Setting Up Tax Rates" for more information.

3. Configure pipelines to apply taxes during pipeline batch rating.

   See "Setting Up Pipelines to Tax Events" for more information.

You can also configure BRM to apply taxes during the billing process. See "Enabling Taxation During Real-Time Rating or Billing" for more information.

Enabling Taxation in Pipeline Rate Plans

When you create pipeline rate plans in Pricing Center, you specify whether events rated by those rate plans should be taxed.

To enable taxation for events in a pipeline rate plan:

1. Select **View - Pipeline Toolbox** and then click **Rate Plan**.
2. Double-click an existing rate plan, or click **New** to create a new rate plan.
3. In the Rate Plan dialog box, select the **Tax Treatment** option as shown in Figure 2–1.

Figure 2–1 Rate Plan Dialog Box

For more information on how to enable taxation in pipeline rate plans, see Pricing Center Help.

Setting Up Tax Rates

To set up tax rates, perform the following:

- Defining Tax Information in Pricing Center
- Configuring Tax Rates in the Tax Code Mapping File

Defining Tax Information in Pricing Center

Follow these steps to define tax information in Pricing Center:

1. Define your tax codes:
   a. Select **View - Pipeline Setup Toolbox**.
   b. In the **Pipeline Setup Toolbox**, select **Financial - Tax Code**.
   c. Double-click an existing tax code, or click **New** to create a new tax code.
   d. In the Tax Code dialog box, specify a tax code name, tax group, tax rate, and validity period.
2. Define your tax groups:
   a. In the **Pipeline Setup Toolbox**, select **Financial - Tax Group**.
   b. Double-click an existing tax group, or click **New** to create a new tax group.
   c. In the Tax Group dialog box, specify a tax group name, tax code, tax rate, and validity period.
3. Associate your tax code with a general ledger (G/L) account:
   a. In the **Pipeline Setup Toolbox**, select **Financial - GL Account**.
   b. Double-click an existing G/L account, or click **New** to create a new GL account.
   c. In the GL Account dialog box, select a tax code in the **Tax Code** field.
4. Associate your G/L account with a price model:
   a. Select **View - Pipeline Toolbox** from the Pricing Center menu.
   b. In the **Pipeline Toolbox**, select **Price Model**.
   c. Double-click an existing price model, or click **New** to create a new price model.
   d. In the Price Model dialog box, click the **Price Model Step** tab.
   e. Select an existing price model step and click **Edit**, or click **New** to create a new step.
   f. In the Price Model Step dialog box, select a G/L account from the **GL Account** list.

See Customer Center Help for more information on setting up data for pipeline rating.

Configuring Tax Rates in the Tax Code Mapping File

To map event criteria to specific tax rates:

1. Open the tax codes mapping file (*BRM_Home*/**sys/cm/taxcodes_map**) in a text editor.
2. Add entries for any custom flat taxes.

   BRM uses these tax rates during the pipeline batch rating process. See "Calculating Flat Taxes by Using the taxcodes_map File" for more information.

   For example, the following entry specifies a custom flat tax of 4.5% for all calls made to Great Britain between February 1, 2004 and January 31, 2005:

```
# Taxcode   Pkg    Rate    Start      End        Lvl    List    Descr    Rule
# -------   ---    ----    --------   --------   ---    ----    -----    ----
  NORM    : U    : 4.5   : 02/01/04 : 01/31/05 : Fed  : GB    : VAT-G  : Std
```

3. Add tax code entries for any third-party tax software.

   BRM uses these codes during the billing cycle only. See "About Specifying Tax Codes" for more information.

```
#   Taxcode           Pkg   Code1   Code2   Si   Wt_code   Cmdty_code
#   ---------------   ---   -----   -----   --   -------   ----------
    usage           : T  :  85000 : 85000 :  S   :         :
```

4. Stop and restart your pipelines to apply your changes.

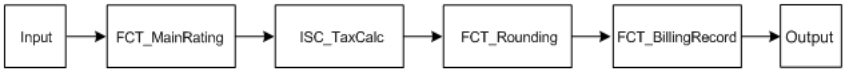
Setting Up Pipelines to Tax Events

You can configure a pipeline to apply flat taxes to events. To apply flat taxes during pipeline batch rating, you configure ISC_TaxCalc in your pipelines.

> **Note:** A pipeline can apply only flat taxes. To use more complex taxation, you must defer taxation to the billing cycle.

ISC_TaxCalc must run *after* the FCT_MainRating module and *before* the FCT_BillingRecord module. If you want to round the tax balance impact, you can optionally run the FCT_Rounding module immediately after ISC_TaxCalc as shown in Figure 2–2.

Figure 2–2 Pipeline Taxation Process with FCT_Rounding



Pipeline Manager applies taxes as follows:

1. The pipeline input and function modules process call records.
2. The FCT_MainRating module rates the call record and adds charge packets to the associated charge breakdown record. See "FCT_MainRating" module in *BRM Configuring Pipeline Rating and Discounting*.
3. The ISC_TaxCalc iScript module applies a flat tax. See "ISC_TaxCalc" iScript module in *BRM Configuring Pipeline Rating and Discounting*

See "About Applying a Flat Tax by Using the ISC_TaxCalc iScript" for more information.

4. The FCT_BillingRecord module consolidates the billing charges and flags whether an event was taxed in a pipeline. See "FCT_BillingRecord" module in *BRM Configuring Pipeline Rating and Discounting*.

   See "About Consolidating Tax Data by Using the FCT_BillingRecord Module" for more information.

5. (Optional) The FCT_Rounding module rounds tax balance impacts. See "FCT_Rounding" module in *BRM Configuring Pipeline Rating and Discounting*

   See "Configuring Resource Rounding" in *BRM Setting Up Pricing and Rating*.

6. The remaining function and output modules finish processing the call records and generate output files.

See "Configuring Pipelines to Apply Flat Taxes" for information about configuring the pipeline for taxation.

Configuring Pipelines to Apply Flat Taxes

To configure your pipelines to apply flat taxes:

1. Use ISC_TaxCalc in your pipelines.

   See "About Applying a Flat Tax by Using the ISC_TaxCalc iScript" and "About Consolidating Tax Data by Using the FCT_BillingRecord Module" for more information.

2. Configure ISC_TaxCalc. See "ISC_TaxCalc" in *BRM Configuring Pipeline Rating and Discounting*.

About Applying a Flat Tax by Using the ISC_TaxCalc iScript

You use the ISC_TaxCalc iScript to apply flat taxes to charges that are flagged for taxation.

When ISC_TaxCalc processes an event data record (EDR), it determines whether the EDR should be taxed by checking whether the **Tax Treatment** flag was applied to the pipeline rate plan that applies to the event. If the flag isn't set, ISC_TaxCalc ignores the EDR and passes it on to the next module in the pipeline. If the Tax Treatment flag is set, ISC_TaxCalc:

- Determines which tax rate to use by checking the **taxcodes_map** data stored in the cache:
  - If the event meets one of the tax criteria, ISC_TaxCalc uses the appropriate tax rate defined in the **taxcodes_map** file.
  - If the event does not meet any tax criteria, ISC_TaxCalc uses the default tax rate specified in the registry file.
- Calculates the flat tax for the pre-tax amount.

  Note: The pre-tax amount is the charged amount minus any discounts.

- Populates the tax packet data block in the EDR with the pre-tax amount, tax percentage, and tax balance impact.

Note: You can round the tax balance impact by running the "FCT_Rounding" module in *BRM Configuring Pipeline Rating and Discounting* immediately after ISC_TaxCalc.

About Consolidating Tax Data by Using the FCT_BillingRecord Module

You use the FCT_BillingRecord module to consolidate charge packets, discount packets, and tax packets into an associated BRM billing record in the EDR.

When an EDR is flagged for taxation, FCT_BillingRecord:

- Creates a balance impact packet that consolidates the charge packet and any tax packet data.
- Flags whether ISC_TaxCalc applied taxes to the EDR by using the PIN_DEFERRED_AMOUNT field. It sets this field to **0** when an EDR contains a tax packet and to PIN_AMOUNT when an EDR does not contain a tax packet.

Sample Flat Tax Configurations

This section shows how to set up flat taxes for two sample applications:

- Applying Flat Taxes to Outcollect Roaming Calls
- Applying Flat Taxes to Third-Party Content Usage

Applying Flat Taxes to Outcollect Roaming Calls

Roaming allows customers of one network operator to use their mobile phones in foreign networks. When customers travel to other regions, they can use the services of any network operator that has a roaming agreement with their home network operator.

When customers from another network use your network (outcollect roaming), you rate those calls and bill the customer's network operator for this usage. Applying a flat tax during the rating process allows you to pass on local taxes to the other network operator.

To apply a local flat tax to outcollect roaming calls:

1. In Pricing Center, ensure that the **Tax Treatment** flag is set in your pipeline rate plans.

   See "Enabling Taxation in Pipeline Rate Plans" for more information.

2. Set up a custom flat tax.

   See "Setting Up Tax Rates" for more information.

3. Configure the ISC_TaxCalc iScript in your outcollect processing pipeline.

   See "Configuring Pipelines to Apply Flat Taxes" for more information.

After the pipeline generates output files that include both usage and tax charges:

- Send one output file to the other network operator for verification and rating.
- Load settlement data into the BRM database and generate invoices for billing the other network operator.

For more information about outcollect roaming, see "About Roaming Outcollect Processing" in *BRM Configuring Roaming in Pipeline Manager*.

Applying Flat Taxes to Third-Party Content Usage

Wireless carriers allow customers to access third-party content, such as news and sports scores, through their mobile phones. Customers are often charged monthly subscription fees to access the content, and usage fees when they download files. Carriers bill their customers for this usage and then remit a portion of the fees to the third-party provider.

In this scenario, taxes are applied to:

- Usage events during the pipeline batch rating process.
- Cycle and one-time purchase events during the billing process.

To configure your system to apply flat taxes to third-party content usage:

1. In Pricing Center, ensure that the **Tax Treatment** flag is set in your pipeline rate plans.

   See "Enabling Taxation in Pipeline Rate Plans" for more information.

2. Set up your custom tax rates.

   See "Setting Up Tax Rates" for more information.

3. Configure the ISC_TaxCalc iScript in your pipelines.

   See "Configuring Pipelines to Apply Flat Taxes" for more information.

4. Use Rated Event (RE) Loader to load your pipeline output files into the BRM database.

   See "Understanding Rated Event Loader" in *BRM Configuring Pipeline Rating and Discounting*.

5. Configure BRM to perform deferred taxation.

   See "About Specifying When to Tax Events" for more information.

When BRM completes the billing process, you can run the remittance utility and then generate a report that summarizes the amount owed to each third-party provider. For more information, see "Remitting Funds to Third Parties" in *BRM Configuring and Running Billing*.

3

Setting Up Tax Calculation for Vertex

This chapter describes how to set up your Oracle Communications Billing and Revenue Management system to use Vertex. BRM supports the following Vertex tax packages:

- Vertex Sales Tax Q Series: Calculates sales and use taxes.
- Vertex Communications Tax Q Series: Calculates telecommunication taxes.

To use Vertex Sales Tax Q Series or Vertex Communications Tax Q Series with BRM, you must install one of the following:

- Vertex Manager: Supports Vertex Sales Tax Q Series and Vertex Communications Tax Q Series.
- Vertex Quantum Manager: Supports Vertex Sales Tax Q Series. This manager does *not* support Vertex Communications Tax Q Series.

> **Important:** Each of these managers is an optional component that you install separately.

See "About Calculating Taxes" for more information about calculating taxes.

See "Customizing Tax Calculation" for information about customizing the Vertex Data Manager (DM).

About Implementing Tax Calculation

To implement Vertex tax calculation, perform the following tasks:

- Installing Vertex Manager or Vertex Quantum Manager
- Installing the Vertex Software
- Configuring the Vertex Software
- Configuring the Vertex DM
- Setting up Tax Supplier Information
- Providing Entries for Vertex Tax Codes in the taxcodes_map File
- Configuring the CM for Vertex Tax Calculation
- Verifying That Taxes Are Being Calculated

Installing Vertex Manager or Vertex Quantum Manager

To install Vertex Manager or Vertex Quantum Manager, see "Installing Tax Calculation Managers" in *BRM Installation Guide* for more information.

Installing the Vertex Software

Install the Vertex Sales Tax Q Series and Communications Tax Q Series software on the same machine as the Vertex Data Manager (DM). The Vertex software must be installed, configured, and tested as indicated in the Vertex documentation before any attempt is made to configure the Vertex DM.

See the Vertex documentation for installation instructions. You can install either package or both.

> **Note:** The Sales Tax Q Series and Communications Tax Q Series database can reside in the same tablespace as the BRM database, but it is recommended that you create a new tablespace for it.

Write down the values you entered for the following when you installed the Vertex Sales Tax Q Series.

- Datasource For Register Database
- Server Name For Register Database
- User Id For Register Database
- Password For Register Database

These values are used later when you configure the **pin.conf** entries.

Configuring the Vertex Software

To configure the Vertex software, set up the Sales Tax Q Series or Communications Tax Q Series shared libraries as described in the following sections:

- Configuring the Sales Tax Q Series Shared Libraries
- Configuring the Communications Tax Q Series Shared Libraries

Configuring the Sales Tax Q Series Shared Libraries

Make sure the following libraries are in the *BRM_Home*/**lib** directory or in your **$LD_LIBRARY_PATH**. These are shared libraries that come with the Sales Tax Q Series software.

Use files with extension **.so** for HP-UX IA64, Linux, and Solaris:

- **libvst**
- **libloc**
- **libqutil**

> **Note:** For AIX, change the extension of these files to **.a**.

For more information about using Vertex toolkits, see the Vertex documentation.

See "About Tax Suppliers" for information about tax suppliers.

Specifying Storage Manager Shared Library for Sales Tax Q Series

Table 3–1 lists the pin.conf entries in the *BRM_Home*/**sys/dm_vertex/pin.conf** file used to configure the Vertex DM for Sales Tax Q Series:

Table 3–1 pin.conf Entries for Vertex DM (Sales Tax Q Series)

Entry	Description
quantumdb_source	The schema where the STQ tables reside. **Note:** This parameter is required if the default tablespace for the user name and password parameters is not the same as the one that contains the STQ tables. For Indexed Sequential Access Method (ISAM) databases, this parameter specifies the ISAM data file directory.
quantumdb_server	The network identifier for the database on the server. For ISAM databases, this parameter should be commented out.
quantumdb_user	The user as a valid Oracle login name. For ISAM databases, this parameter should be commented out.
quantumdb_passwd	The user password.
quantum_sm_obj	The Storage Manager shared library that the DM uses to interact with a particular version of the Vertex STQ system. To use ■ STQ 3.1.6, specify **dm_vertex_stq316** ■ STQ 3.2.21, specify **dm_vertex_stq3221** ■ STQ 4.0.6, specify **dm_vertex_stq406**

To specify the appropriate Storage Manager shared library, set the value for **quantum_sm_obj** entry in the pin.conf file for Vertex DM to the particular version of Vertex STQ that you intend to use:

1. Open the Vertex DM configuration file (*BRM_Home*/**sys/dm_vertex/pin.conf**).
2. Locate the **quantum_sm_obj** entry displayed in the following format in the file:

   ```
   - dm_vertex quantum_sm_obj ./dm_vertex_stqVersion${LIBRARYEXTENSION}
   ```

   where *Version* represents the STQ version.

   For example, the following entry specifies that the Vertex DM needs to use STQ 4.0.6.

   ```
   - dm_vertex quantum_sm_obj ./dm_vertex_stq406${LIBRARYEXTENSION}
   ```

3. Save and close the file.
4. Stop and restart the Vertex DM.

   See the discussion about starting and stopping the BRM system in *BRM System Administrator's Guide* for more information.

Configuring the Communications Tax Q Series Shared Libraries

Make sure the following libraries are in the *BRM_Home*/**lib** directory or in your **$LD_LIBRARY_PATH**. These are shared libraries that come with the Vertex Communications Tax Q Series software. Use files with extension **.so** for HP-UX IA64, Linux, and Solaris:

- **libadm**
- **libcch**
- **libcfg**
- **libcli**
- **libctq** (For AIX *only*, change the extension of this file to **.a**.)
- **libctz**
- **libdbcper**
- **libgeo**
- **libhsh**
- **libobj**
- **libreg**
- **librpt**
- **librte**
- **libutl**
- **libxmlparse**

Specifying Storage Manager Shared Library for Communications Tax Q Series

Table 3–2 lists the entries in the *BRM_Home*/**sys/dm_vertex/pin.conf** file used to configure the Vertex DM for Communications Tax Q Series:

Table 3–2 pin.conf Entries for Vertex DM (Communications Tax Q Series)

Entry	Description
commtax_config_path	The location of Communications Tax Q Series configuration file (**ctqcfg.xml**). The default location is *CTQ_Home*/**vertex/cfg**, where *CTQ_Home* is the directory where you installed the Vertex software.
commtax_config_name	The Communications Tax Q Series configuration name. This name must match the configuration defined in the **ctqcfg.xml** file that is used with BRM. See the Vertex documentation for more information about defining configurations.
commtax_sm_obj	The Storage Manager shared library that the DM uses to interact with a particular version of the Vertex CTQ system. To use ■ CTQ 1.01.06, specify **dm_vertex_ctq10106** ■ CTQ 1.00.13, specify **dm_vertex_ctq10013** ■ CTQ 2.00.05, specify **dm_vertex_ctq20005**

To specify the appropriate Storage Manager shared library, set the value for **commtax_sm_obj** entry in the pin.conf file for Vertex DM to the particular version of Vertex CTQ that you intend to use:

1. Open the Vertex DM configuration file (*BRM_Home*/**sys/dm_vertex/pin.conf**).
2. Locate the **commtax_sm_obj** entry displayed in the following format in the file:

```
dm_vertex commtax_sm_obj ./dm_vertex_ctqVersion${LIBRARYEXTENSION}
```

where *Version* represents the CTQ version.

For example, the following entry specifies that the Vertex DM needs to use CTQ 2.00.05.

```
dm_vertex commtax_sm_obj ./dm_vertex_ctq20005${LIBRARYEXTENSION}
```

3. Save and close the file.
4. Stop and restart the Vertex DM.

   See the discussion of starting and stopping the BRM system in *BRM System Administrator's Guide*, for more information.

Configuring the Vertex DM

You use the Vertex DM configuration file (*BRM_Home*/**sys/dm_vertex/pin.conf**) to configure the DM. The file includes information for editing its contents. See "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide* for information about configuration files.

The following procedures provide specific information about configuring the Vertex DM:

- Configuring the Vertex DM for Sales Tax Q Series
- Configuring the Vertex DM for Communications Tax Q Series

> **Note:** Record the port and database numbers entered in your Vertex DM **pin.conf** file. These entries are also used in the CM **pin.conf** file.

Configuring the Vertex DM for Sales Tax Q Series

You specify the required **pin.conf** entries in the *BRM_Home*/**sys/dm_vertex/pin.conf** file to configure the Vertex DM for Sales Tax Q Series:

1. Open the Vertex DM configuration file (*BRM_Home*/**sys/dm_vertex/pin.conf**).
2. Change the following entries according to the instructions in the file:

   ```
   - dm_vertex quantumdb_source tablespace_for_Sales_Tax_C_Series_tables
   - dm_vertex quantumdb_server database_name
   - dm_vertex quantumdb_user database_user
   - dm_vertex quantumdb_passwd database_user_password
   ```

   where:

   - the values for **quantumdb_source**, **quantumdb_server**, **quantumdb_user**, and **quantumdb_passwd** are the same values used when you installed the Sales Tax Q Series software. See "Installing the Vertex Software".
   - **quantumdb_source** is required if the default tablespace for the user ID and password parameters is not the same tablespace that contains the Sales Tax Q Series tables.

Important: If the Vertex Sales Tax Q Series installation uses an ISAM database:

- **quantumdb_source** should point to the ISAM data file directory (as opposed to the Oracle data source).
- **quantumdb_server**, **quantumdb_user**, and **quantumdb_passwd** should be commented out.
- **dm_n_fe** and **dm_n_be** should be set to **1** to prevent multi-threading. The ISAM version of Vertex does not support multi-threading.

3. Save and close the file.
4. Stop and restart the Vertex DM.

   See the discussion of starting and stopping the BRM system in *BRM System Administrator's Guide*, for more information.

Configuring the Vertex DM for Communications Tax Q Series

You set the required **pin.conf** entries in the *BRM_Home***/sys/dm_vertex/pin.conf** file to configure the Vertex DM for Communications Tax Q Series:

To configure the Vertex DM for Communications Tax Q Series:

1. Open the Vertex DM configuration file (*BRM_Home***/sys/dm_vertex/pin.conf**).
2. Specify the path to the Communications Tax Q Series configuration file in the **commtax_config_path** entry.

   The following entry specifies the default path:

   ```
   - dm_vertex commtax_config_path CTQ_Home/vertex/cfg
   ```

3. Specify the Communications Tax Q Series configuration name in the **commtax_config_name** entry.

   For example:

   ```
   - dm_vertex commtax_config_name CTQ_Test
   ```

4. Specify the category codes that use ChargeTo as the primary place of use (PPU) location.

   For example:

   ```
   - dm_vertex use_charge_to_category_codes 12,40,10
   ```

5. Save and close the file.
6. Stop and restart the Vertex DM.

   See the discussion of starting and stopping the BRM system in *BRM System Administrator's Guide*, for more information.

Setting up Tax Supplier Information

To set up tax suppliers in the BRM database:

- Define at least one tax supplier (**/config/tax_supplier** object).

See "Defining Tax Suppliers" for more information.

- Edit the **tax_supplier_map** file.

  See "Providing Tax Supplier Data in a tax_supplier_map File" for more information.

- Configure the **taxcodes_map** file.

  See "Providing Entries for Vertex Tax Codes in the taxcodes_map File" for more information.

Defining Tax Suppliers

To define tax suppliers, edit the *BRM_Home***/sys/data/config/pin_tax_supplier.xml** file. See "Defining Tax Suppliers and Loading Them into the BRM Database" for more information.

Providing Tax Supplier Data in a tax_supplier_map File

You provide tax supplier data in a **tax_supplier_map** file that you edit. See "Allowing Your Sales and Use Tax Software to Determine Tax Supplier Data" for more information.

Table 3–3 provides a list of the entries in the **tax_supplier_map** file:

Table 3–3 Entries in tax_supplier_map File

Entry	Description
Product	The name of the product for which taxes will be calculated. Product names are defined in your price list. For more information, see "Setting Up Price List Data" in *BRM Setting Up Pricing and Rating*.
ShipTo	The address the product is shipped to (your customer's location). Enter the address in this format: *city*;*state*;*ZIP*;*country* You can leave portions of this string blank, but you must include the semicolons for each address element. Examples: ■ ;;;US ■ ;CA;95003;US In most cases, you do not need to enter the city. See the Vertex documentation for correct state and country entries.
Company ID	The company ID specified in the Vertex TDM (Tax Decision Maker) or company file. This name is also entered in the PIN_FLD_NAME field when you define tax suppliers. It is also displayed in Pricing Center. This entry must match the name exactly as entered in the TDM or company file. If you use the TDM, the entry will be a four-character code.
Business Loc	The business location specified in the Vertex TDM file or company file. This value indicates which location should be considered when taxes are calculated.

Table 3–3 (Cont.) Entries in tax_supplier_map File

Entry	Description
ShipFrom	The address the product is shipped from. Enter the address in this format: *city*;*state*;*ZIP*;*country* You can leave portions of this string blank, but you must include the semicolons for each address element. Examples: ■ ;;;US ■ ;CA;95003;US See the Vertex documentation for correct state and country entries.
Regulated	The flag that identifies the utility or company that is billing as regulated (**0**) or unregulated (**1**). Some taxes apply to regulated or unregulated utilities only; this flag determines which types of taxes are applied.

To create the **tax_supplier_map** file:

1. Provide the necessary information on each tax supplier. See Table 3–3 for an explanation of the **tax_supplier_map** file.

   For example:

```
Product     ShipTo  Company ID      Business Loc  ShipFrom              Reg
_____________________________________________________________________________
user_code : ;;;US : Tax supplier1 : loc_1       : Cupertino;CA;95014;US :0
usage     : ;;;US : Tax supplier1 : loc_1       : Cupertino;CA;95014;US :0
usage     : ;;;FR : Tax supplier2 : loc_2       : Paris;;;FR            :1
```

2. Save and close the file.
3. Edit the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**) to supply the location of the **tax_supplier_map** file.

   If you create a **tax_supplier_map** file, you need to include its location in the CM configuration file. See "Specifying the Location of the tax_supplier_map File" for more information.

Specifying Divisions for Tax Suppliers

Some companies operate divisions with locations that can impact the tax calculation. Use the **Business Loc** column in the **tax_supplier_map** file to add information about divisions for tax suppliers. The data in this column is mapped to the **Division** field in Sales Tax Q Series. This data should match what is stored in the Sales Tax Q Series TDM database.

For example, suppose the tax supplier Acme has two divisions, one in California and one in Illinois. You can configure the **tax_supplier_map** file to assign customers to the appropriate location. For example:

```
# Product    ShipTo    Company Id  Business Loc   ShipFrom              Regulated
# -------    ------    ----------  ------------   --------              ---------
electrical : ;;CA;US : Acme      : West         : Cupertino;CA;95014;US : 1
electrical : ;;TX;US : Acme      : Central      : Oak Brook;IL;60523;US : 1
```

With this configuration, for customers in California, BRM sends **West** as the division to Sales Tax Q Series. The tax supplier's address will be Cupertino, CA. Similarly,

customers in Texas will be assigned to the **Central** division with the Oak Brook, IL address.

Providing Entries for Vertex Tax Codes in the taxcodes_map File

See "About Specifying Tax Codes" for information about tax codes.

To provide the Vertex entries in the **taxcodes_map** file, you need the following information:

- The names you will use for tax codes.
- The corresponding Vertex product codes.
- Whether the product is for sale (S) or resale (R).

> **Note:** **Wt_code** and **Cmdty_code** do not apply to Vertex.

For information about Vertex product codes and service indicators, see the Vertex documentation.

To provide Vertex tax codes in the **taxcodes_map** file:

1. Go to the directory where the **taxcodes_map** file is located. The default location is *BRM_Home***/sys/cm**.
2. Open the **taxcodes_map** file.
3. Edit the file. For example:

```
#   Taxcode           Pkg   Code1   Code2   Si   Wt_code   Cmdty_code
#   ---------------   ---   -----   -----   --   -------   ----------
#   usage2          :  Q  :    01 :    01 :  S  :        :
#   direct          :  C  :    01 :    01 :  S  :        :
#   toll            :  C  :    04 :    02 :  S  :        :
#   direct2         :  B  :    01 :    04 :  S  :        :
#   toll2           :  B  :    01 :    01 :  S  :        :
#   user_code       :  U  :  10.0
monthlyCard         :  Q  :    03 :    03 :  S  :        :
installVertex       :  Q  :    03 :    03 :  S  :        :
interCommtax        :  C  :    04 :    01 :  S  :        :
intraCommtax        :  C  :    01 :    02 :  S  :        :
ipUsage             :  Q  :    03 :    03 :  S  :        :
```

   For an explanation of an entry, see the **taxcodes_map** file.

4. Save and close the file.
5. Stop and restart the CM.

   See the discussion of starting and stopping the BRM system in *BRM System Administrator's Guide*, for more information.

Configuring the CM for Vertex Tax Calculation

You configure the Connection Manager (CM) for Vertex tax calculation by editing the CM configuration file (**pin.conf**). For more information on editing configuration files, see "Using Configuration Files to Connect and Configure Components" in *BRM System Administrator's Guide*.

Configure CM for Vertex calculation by completing these tasks:

- Specifying Vertex DM Connection Entries in CM Configuration File
- Specifying whether to Validate ZIP Codes
- Defining a Default Ship-From Locale
- Specifying How to Calculate Taxes
- Specifying the Location of the tax_supplier_map File
- Specifying the Location of the taxcodes_map File
- Itemizing or Summarizing Taxes for Each Jurisdiction Level

Specifying Vertex DM Connection Entries in CM Configuration File

When you install BRM, you specify connection entries. You must change these entries if you change the Vertex database number, the host name, or the port number of the Vertex DM.

To specify the Vertex DM connection entry in the CM configuration file:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. Edit the **dm_pointer** entry.

   ```
   - cm  dm_pointer  database  ip  hostname  port
   ```

   where:

   - *database* identifies the Data Manager, for example, 0.0.8.1.
   - *hostname* is the IP address or host name of the computer on which the DM is installed.
   - *port* is the port number of the DM service.

   Note: The database number, host name, and port number must match the values in the Vertex DM configuration file, *BRM_Home*/**sys/dm_vertex/pin.conf**.

3. Edit the **vertex_db** entry.

   ```
   - fm_rate  vertex_db  database  /_tax_db  0
   ```

   where *database* is the database number specified in the Vertex DM configuration file.

4. Save and close the file.

You don't need to restart the CM to enable these entries.

Specifying whether to Validate ZIP Codes

If there is an error in an account's ZIP code, Sales Tax Q Series returns a tax amount of 0. If you set up Sales Tax Q Series to validate ZIP codes at registration, you ensure that taxes are calculated correctly. This option specifies that Sales Tax Q Series checks the city, state and ZIP code when an account is created. If the ZIP code is not valid, customer registration cannot be completed.

Important:

- Communications Tax Q Series does not validate addresses and always returns a valid result.
- If you enable this option, the Vertex DM must be running when customers register. If the connection to Vertex is offline, you can change this option to skip validating ZIP codes. This allows customers to register.

To specify whether to validate ZIP codes:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. Change the **tax_valid** entry:
   - To enable ZIP code validation, enter **3**.
   - To disable ZIP code validation, enter **0**. This is the default setting.

   See "About Validating Customer Addresses" for more information.
3. Save and close the file.

You don't need to restart the CM to enable this entry.

Defining a Default Ship-From Locale

This option provides a ship-from locale in case a tax supplier cannot be found. You need to define at least one tax supplier. See "About Tax Suppliers" for more information.

To define a default ship-from locale:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. Edit the **provider_loc** entry. For example:

```
- fm_rate_pol provider_loc Middletown, CA 95222 USA
```

3. Save and close the file.

The new value becomes effective immediately and all subsequent tax calculations use the new address. You do not need to restart the CM.

Specifying How to Calculate Taxes

You use the **cycle_tax_interval** entry to determine whether deferred taxes are calculated separately for a parent and its subordinate (nonpaying) child accounts or are consolidated into a single tax item for both the parent and child accounts. See "About Tax Calculation for Account Groups" for more information.

Important: Regardless of the option you select for the entry, Bill Now always rolls activities for each subordinate bill unit into the parent bill unit and calculates taxes for the parent only. The single tax item for the parent includes taxes from both the parent and subordinate bill units.

To specify how to calculate taxes:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).

2. Edit the **cycle_tax_interval** entry:
   - To forward the tax from the child account to the parent account, enter **billing**. BRM calculates taxes for the parent account only, but the single tax item on the parent account includes taxes from both the parent and child accounts.
   - To calculate the taxes separately for the parent and child accounts, enter **accounting**. BRM lists the taxes as separate items on the parent bill.

   For example:

   ```
   fm_bill  cycle_tax_interval  billing
   ```

3. Save and close the file.
4. Stop and restart the CM.

   See the discussion of starting and stopping the BRM system in *BRM System Administrator's Guide*, for more information.

Specifying the Location of the tax_supplier_map File

To specify the location of the **tax_supplier_map** file:

1. Open the CM configuration file (*BRM_Home***/sys/cm/pin.conf**).
2. Add the **tax_supplier_map** entry. For example:

   ```
   - fm_rate   tax_supplier_map  BRM_Home/sys/cm/tax_supplier_map
   ```

3. Save and close the file.
4. Stop and restart the CM.

   See the discussion of starting and stopping the BRM system in *BRM System Administrator's Guide*, for more information.

Specifying the Location of the taxcodes_map File

You use the **taxcodes_map** file to map tax codes to Vertex products. See "About Specifying Tax Codes" for more information.

To specify the location of the **taxcodes_map** file:

1. Open the CM configuration file (*BRM_Home***/sys/cm/pin.conf**).
2. Edit the **taxcodes_map** entry to specify the complete path to the **taxcodes_map** file. The default location is *BRM_Home***/sys/cm**.)

   For example:

   ```
   - fm_rate   taxcodes_map  BRM_Home/sys/cm/tax_supplier_map
   ```

3. Save and close the file.
4. Stop and restart the CM.

   See the discussion of starting and stopping the BRM system in *BRM System Administrator's Guide*, for more information.

Itemizing or Summarizing Taxes for Each Jurisdiction Level

You can opt to show the details of the tax types returned by Communications Tax Q Series for each tax jurisdiction level. These details can be mapped to a general ledger (G/L) ID and be shown on the invoice.

To itemize the taxes for each jurisdiction:

1. Open the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**).
2. Add the following entry:

   ```
   - fm_rate tax_return_juris itemize
   ```

   To summarize the taxes, change the entry to **summarize** instead of **itemize**.
3. Save and close the file.
4. Stop and restart the CM.

   See the discussion of starting and stopping the BRM system in *BRM System Administrator's Guide*, for more information.

Numbering Differences in Vertex-to-BRM Jurisdiction Code Mapping

Some of the Vertex CTQ 2.00.05 and later jurisdiction codes are different from the internal BRM jurisdiction codes to which they are mapped. Table 3–4 lists the Vertex-to-BRM mappings whose codes differ:

Table 3–4 Numbering Differences in Vertex-to-BRM Jurisdiction Code Mapping

Jurisdiction	Vertex Code	BRM Code
Other Municipality	6	12
County District	7	10
City District	9	11

Important: You must use the new BRM jurisdiction codes (12, 10, and 11) while specifying the account's tax exemptions and also while interpreting the jurisdiction codes in the **/event** object.

Verifying That Taxes Are Being Calculated

After you have installed and configured the Vertex software, you can run the following test to verify that the Vertex software is calculating taxes.

Important: The PIN_FLD_TAX_CODE value determines which tax package to use. This value is defined in the **taxcodes_map** file and is required to calculate taxes successfully. See "Providing Entries for Vertex Tax Codes in the taxcodes_map File" for more information.

1. Create a text file named **T502_40** with the following contents:

```
0 PIN_FLD_POID                    POID [0] 0.0.0.1 /account 1 1
0 PIN_FLD_END_T                  TSTAMP [0] (938224941) Fri Sep 24 19:02:21 1999
0 PIN_FLD_ACCOUNT_NO                STR [0] "ROOT"
0 PIN_FLD_CURRENCY                  INT [0] 840
0 PIN_FLD_CURRENCY_NAME             STR [0] "USD"
0 PIN_FLD_TAXES                   ARRAY [0] allocated 20, used 10
1     PIN_FLD_TAX_CODE              STR [0] "installVertex"
1     PIN_FLD_AMOUNT_TAXED      DECIMAL [0] 20
1     PIN_FLD_GL_ID                 INT [0] 0
1     PIN_FLD_SHIP_TO               STR [0] "Cupertino;CA;95014;US;[408572,2,1]"
1     PIN_FLD_SHIP_FROM             STR [0] "Denver; CO; 80205;US;[303279,2,1]"
```

```
1     PIN_FLD_ORDER_ORIGIN         STR [0] ""
1     PIN_FLD_ORDER_ACCEPT         STR [0] ""
1     PIN_FLD_INTERNATIONAL_IND    INT [0] 1
1     PIN_FLD_LOCATION_MODE       ENUM [0] 2
1     PIN_FLD_ELAPSED_TIME      TSTAMP [0] (249)
0 PIN_FLD_VAT_CERT                 STR [0] ""
0 PIN_FLD_INCORPORATED_FLAG       ENUM [0] 0
0 PIN_FLD_RESIDENCE_FLAG          ENUM [0] 0
0 PIN_FLD_REGULATED_FLAG          ENUM [0]
```

2. Save the text file to the *BRM_Home*/**setup/scripts** directory.
3. Start **dm_vertex**.
4. Open the *BRM_Home*/**sys/cm.pinlog** file and verify there are no errors.
5. Run **testnap** in the *BRM_Home*/**setup/scripts** directory to verify that the taxes are being calculated:

```
testnap
r T502_40 1
xop 502 0 1
```

Note: 502 is the opcode reference number for PIN_FLD_RATE_TAX_CALC.

4

Customizing Tax Calculation

This chapter describes how to extend the Oracle Communications Billing and Revenue Management (BRM) tax calculation features, including the Vertex Data Manager.

See "About Calculating Taxes" for information on calculating taxes.

How BRM Calculates Taxes

The PCM_OP_RATE_EVENT opcode is the main tax calculation opcode.

For each rated event, this opcode reads the PIN_FLD_RATES_USED array in the input flist to determine what rates to apply. This rate information determines how the taxable amount is treated.

Based on the information it collects, PCM_OP_RATE_EVENT does the following:

- If taxation occurs during rating, it calls the PCM_OP_RATE_TAX_CALC opcode.
- If taxation occurs during billing, it calls the PCM_OP_BILL_CYCLE_TAX opcode. PCM_OP_BILL_CYCLE_TAX calls PCM_OP_RATE_TAX_CALC to perform the tax calculation.
- If the event is nontaxable, it does nothing.

PCM_OP_RATE_EVENT also calls the PCM_OP_RATE_POL_TAX_LOC policy opcode to determine whether custom processing or default processing is used to determine the tax-related locales.

- If default processing is used, PCM_OP_RATE_EVENT retrieves the locales from the database.
- If custom processing is used, control is handed to the PCM_OP_RATE_POL_TAX_LOC policy opcode, which determines which locales to use.

The locales are used during the tax calculation process to determine jurisdiction. BRM stores up to four tax locales for the current account, session, and event.

PCM_OP_RATE_EVENT also checks the PIN_FLD_EXEMPTIONS array in **/account** objects to determine whether all or part of the purchase amount is exempt from taxes.

PCM_OP_RATE_EVENT returns a revised set of objects for the event. This can include PIN_FLD_TAXES and PIN_FLD_EXEMPTIONS arrays to indicate tax amounts that were calculated immediately by PCM_OP_RATE_TAX_CALC.

See "Retrieving Tax Location Data" for more information on the tax locales and how to customize their retrieval.

Calculating Taxes

PCM_OP_RATE_TAX_CALC performs tax calculations when taxes are calculated during real-time rating or during billing.

- To calculate taxes during real-time rating, PCM_OP_RATE_TAX_CALC is called by PCM_OP_RATE_EVENT.
- To calculate taxes during billing, PCM_OP_RATE_TAX_CALC is called by PCM_OP_BILL_CYCLE_TAX.

PCM_OP_RATE_TAX_CALC determines the taxation Data Manager (DM) or custom taxation method to be used for an event. It sends the input flist to the DM or the custom tax calculation method for calculation. It receives the information it needs to calculate taxes from the following input fields:

- The tax code of the BRM product mapped to a product code that is recognized by the taxation software. The tax code also identifies the taxation DM or the custom tax calculation method to be used.
- The purchase amount to be taxed from the PIN_FLD_TAXES array. Each element of the PIN_FLD_TAXES array represents a separate tax calculation method.

  Note: If the tax calculation software or custom policy is not specified in the tax code, taxes are not calculated.

- The locations that are involved in the transaction to be taxed. This includes ship-from, ship-to, order origin, and order accept addresses for sales and use taxation.

  Note: For telecommunications taxation, the address fields contain additional information (NPA-NXX or geocodes) for origin, termination, and charge-to numbers.

- The date and time of the transaction from the PIN_FLD_START_T and PIN_FLD_END_T fields. PIN_FLD_START_T is used to compute PIN_FLD_ELAPSED_TIME or duration (that is, the difference between the end time and start time).
- Tax exemption information, including exemption information based on tax jurisdiction, from the PIN_FLD_EXEMPTIONS array.
- The tax supplier ID from the PIN_FLD_TAX_SUPPLIER field.
- Other optional values such as the account's bill object in the PIN_FLD_BILL_OBJ field and telecommunications values such as incorporated, residence, and regulated flags.

PCM_OP_RATE_TAX_CALC returns an output flist with a revised PIN_FLD_TAXES array containing the tax data that the tax calculation software or the custom tax calculation method returns. The calculated taxes are sorted by jurisdiction.

Tax Calculation Error Handling

PCM_OP_RATE_TAX_CALC checks for the following errors:

- Missing or nonvalid PIN_FLD_TAX_CODE field.
- Nonvalid PIN_FLD_TAXPKG_TYPE field, which is derived from PIN_FLD_TAX_CODE field and the **taxcodes_map** file.

- Missing or nonvalid tax database entry, such as **vertex_db** in the Connection Manager (CM) configuration file (**pin.conf**).

Tax Calculation Policy Opcodes

PCM_OP_RATE_TAX_CALC calls the following policy opcodes that you can customize to modify the tax data or to use custom tax calculation:

- PCM_OP_RATE_POL_PRE_TAX, to modify any tax data in the input flist that is passed to the taxation DM.

  See "Modifying Tax Data Before Calculating Taxes" for more information.

- PCM_OP_RATE_POL_POST_TAX, to modify any tax data in the output flist that is returned from the taxation DM.

  See "Modifying Tax Data after Calculating Taxes" for more information.

- PCM_OP_CUST_POL_TAX_CALC, to use custom tax calculation methods instead of using an external tax calculation software.

  See "Using Custom Tax Rates" for more information.

See "About Customizing BRM Taxation" for more information about customizing tax calculation.

Calculating Taxes without Recording Them

If the PCM_OPFLG_CALC_ONLY flag is set, PCM_OP_RATE_TAX_CALC performs the tax calculation and returns the tax rate and amount. The tax data is not written to the database, however.

Performing Address or VAT Certificate Validation

You can call PCM_OP_RATE_TAX_CALC in jurisdiction-check-only mode during account creation to perform address or VAT certificate number validation. You specify jurisdiction-check-only mode by using the PIN_FLD_COMMAND input field. The following modes are supported:

- If PIN_FLD_COMMAND is set to **1**, it sends an address to the taxation DM for verification and additionally returns the geocode and county information in the PIN_FLD_NAMEINFO array if the address was successfully validated.
- If PIN_FLD_COMMAND is set to **2**, it sends a VAT certificate number to the tax DM for validation.

In jurisdiction-check-only mode, PCM_OP_RATE_TAX_CALC returns **1** if the jurisdiction is valid and **0** if it is not.

Calculating Taxes during Billing

To calculate taxes during billing, PCM_OP_RATE_EVENT calls PCM_OP_BILL_CYCLE_TAX.

PCM_OP_BILL_CYCLE_TAX performs these steps:

1. If a transaction is not already open, opens one.

   If PCM_OPFLG_CALC_ONLY is set, the transaction is opened with the flag PCM_TRANS_OPEN_READONLY.

2. Creates an **/event/billing/cycle/tax** object with the taxable amount if there were any deferred taxes.

   The output flist contains the POID of the created **/event/billing/cycle/tax** object.

3. Calls PCM_OP_RATE_TAX_CALC to perform the actual tax calculation.

4. If PCM_OP_BILL_CYCLE_TAX was called with the PCM_OPFLG_CALC_ONLY flag set, it still performs the tax calculation but does not create the **/event/billing/cycle/tax** object.

   Instead, this opcode's output flist contains the PIN_FLD_RESULTS array, an flist with the information that would have been used to create the **/event/billing/cycle/tax** object.

5. Sets the **/event/billing/cycle/tax** object status to inactive.

6. Commits the transaction if one was opened in step 1.

About Customizing BRM Taxation

You can customize BRM tax calculation in the following ways:

- To retrieve tax data such as tax codes and tax suppliers from the database. See "Retrieving Tax Calculation Data" for more information.
- To use custom tax rates instead of a taxation package. See "Using Custom Tax Rates" for more information.
- To modify tax data before passing it to the taxation DM. For example, you can set up tax exemptions prior to calculating taxes. See "Modifying Tax Data Before Calculating Taxes" for more information.
- To modify tax data returned from the taxation DM. For example, you can add a surcharge to the tax amount before adding it to the account balance. See "Modifying Tax Data after Calculating Taxes" for more information.
- To use geocodes to identify the taxing jurisdiction for the rated and taxed event. See "Using Geocodes to Calculate Taxes" for more information.

Retrieving Tax Calculation Data

Use the following policy opcodes to retrieve tax data from the BRM database:

- PCM_OP_RATE_POL_GET_TAXCODE

  See "Retrieving a List of Tax Codes" for more information.

- PCM_OP_RATE_POL_GET_TAX_SUPPLIER

  See "Retrieving a List of Tax Suppliers" for more information.

- PCM_OP_RATE_POL_MAP_TAX_SUPPLIER

  See "Retrieving Tax Supplier Data" for more information.

- PCM_OP_RATE_POL_TAX_LOC

  See "Retrieving Tax Location Data" for more information.

Retrieving a List of Tax Codes

To retrieve a list of tax codes, use the PCM_OP_RATE_POL_GET_TAXCODE policy opcode. You can customize this policy opcode to return additional cached tax code information.

The PCM_OP_RATE_POL_GET_TAXCODE policy opcode takes an account POID as input and performs these functions:

- Obtains the database number from the POID.
- Searches the CMs in-memory cache for tax codes.
- Returns an array of tax code names in the output flist.

The PCM_OP_RATE_POL_GET_TAXCODE policy opcode returns a list of the tax code names cached from the **taxcodes_map** file by the CM during initialization.

Retrieving a List of Tax Suppliers

To retrieve a list of tax suppliers, use the PCM_OP_RATE_POL_GET_TAX_SUPPLIER policy opcode. You can customize this policy opcode by modifying the fields on the output flist. You can specify which fields are validated by adding or removing them from the input flist.

The PCM_OP_RATE_POL_GET_TAX_SUPPLIER policy opcode takes an account POID as input and performs these functions:

- Obtains the database number from the POID.
- Performs a global search for tax suppliers.
- Returns a list of tax suppliers with all the relevant fields in the output flist.

See "About Tax Suppliers" for more information on tax suppliers.

Retrieving Tax Supplier Data

To retrieve tax supplier data, use the PCM_OP_RATE_POL_MAP_TAX_SUPPLIER policy opcode. You can customize this policy opcode to change how a tax supplier is derived for a specific BRM event.

This policy opcode can function in either of two ways:

- If a **tax_supplier_map** file is used with a corresponding entry in the CM **pin.conf** file, this policy opcode searches the **tax_supplier_map** file by company ID based on the Product Name and Ship To categories. If a matching entry is found, it returns that entry from the **tax_supplier_map** file. If no matching entry exists, the policy opcode returns **NULL** as the address of the flist.
- If the **tax_supplier_map** file does not exist or the CM **pin.conf** entry does not specify using a **tax_supplier_map** file, this policy opcode tries to find the **tax_supplier** object POID in the PIN_FLD_PRODUCTS array of the account object.

The PCM_OP_RATE_POL_MAP_TAX_SUPPLIER policy opcode also retrieves the utility flag, which is used for telecommunication (telco) taxation, from the PIN_FLD_REGULATED_FLAG field of the **/profile/tax_supplier** object and returns it on the output flist.

Retrieving Tax Location Data

To customize the way PCM_OP_RATE_EVENT obtains the locations for tax jurisdictions, use the PCM_OP_RATE_POL_TAX_LOC policy opcode. This policy

opcode returns the locations for an event, which are then used to establish jurisdictions for tax calculation.

> **Note:** For a telephony event, the locations contain additional information, which is enclosed in square ([]) brackets.

For example, you can obtain the value for the PIN_FLD_ORDER_ACCEPT field from a different source. Additionally, you can provide a geocode instead of an NPA/NXX for telephony events.

The PCM_OP_RATE_POL_TAX_LOC policy opcode takes an account POID as input and returns these fields:

- PIN_FLD_SHIP_TO: The ship-to address; for telephony events, the call termination number.
- PIN_FLD_SHIP_FROM: The ship-from address; for telephony events, the call origination number.
- PIN_FLD_ORDER_ACCEPT: The order accept address; for telephony events, the charge-to number.
- PIN_FLD_ORDER_ORIGIN: The order origin address.

It returns them in four strings with this syntax:

city`;` *state_abbreviation*`;`*zipcode*`;` *country*`;[`*code*`,`*location*`,`*international_indicator*`]`

where:

- *code* is the geocode or NPA/NXX.
- *location* specifies the type of code:

  0 - Address

  1 - Geocode

  2 - NPA/NXX
- *international_indicator* specifies the type of call:

  1 - North America numbering plan

  2 - North America originated, overseas terminated

  3 - North America originated, overseas terminated and billed

  4 - IOC North America originated, overseas terminated

  5 - Overseas originated, North America terminated

  6 - IOC North America originated, overseas terminated and billed

  7 - Overseas originated, overseas terminated

For example:

```
1  PIN_FLD_SHIP_TO      STR [0] "Cupertino; CA; 95014; USA;[408572,2,1]"
1  PIN_FLD_SHIP_FROM    STR [0] "Englewood; CO; 80112; USA;[060050006000,1,1]"
1  PIN_FLD_ORDER_ORIGIN STR [0] "Englewood; CO; 80112; USA;[060050006000,1,1]"
1  PIN_FLD_ORDER_ACCEPT STR [0] "Englewood; CO; 80112; USA;[060050006000,1,1]"
1  PIN_FLD_LOCATION_MODE ENUM [0] 1
1  PIN_FLD_INTERNATIONAL_IND INT [0] 1
```

By default, the PCM_OP_RATE_POL_TAX_LOC policy opcode uses the sources for locations described in Table 4–1:

Table 4–1 PCM_OP_RATE_POL_TAX_LOC Source Locations

Location	Default Source
PIN_FLD_SHIP_TO	The account billing address, obtained from the PIN_FLD_NAMEINFO array of the **/account** object.
PIN_FLD_SHIP_FROM	Same as PIN_FLD_ORDER_ACCEPT.
PIN_FLD_ORDER_ACCEPT	The value for the **fm_rate_pol provider_loc** entry in the CM **pin.conf** file. For information on the default PIN_FLD_ORDER_ACCEPT location, see the **fm_rate_pol provider_loc** entry in the CM **pin.conf** file.
PIN_FLD_ORDER_ORIGIN	Same as PIN_FLD_ORDER_ACCEPT.

Using Custom Tax Rates

If your business uses a relatively uncomplicated tax calculation method, you may not need to use a tax calculation software package. You can use custom tax rates instead. In simple cases, you can implement taxation using built-in features that do not require programming. See "Calculating Flat Taxes by Using the taxcodes_map File" for more information.

If your business requires a more complex tax structure but does not require the full capabilities of one of the taxation packages, you can implement custom tax rates by modifying the two taxation policy opcodes: PCM_OP_CUST_POL_TAX_INIT and PCM_OP_CUST_POL_TAX_CALC.

The PCM_OP_CUST_POL_TAX_INIT policy opcode caches tax rate information from the file specified by the **taxcodes_map** entry in the CM **pin.conf** file. PCM_OP_CUST_POL_TAX_CALC caches the entries in this file in which the **Tax Pkg** value is **U** (user code). It ignores all the other entries including the tax codes Vertex tax calculation software.

If no tax calculation software is configured, the PCM_OP_CUST_POL_TAX_INIT policy opcode calls the PCM_OP_CUST_POL_TAX_CALC policy opcode to use the custom tax rates in the cache instead of calling a taxation DM. The PCM_OP_CUST_POL_TAX_CALC policy opcode performs the actual tax calculation.

To use custom tax rates:

1. Define tax rates in a text file.

   You can use the default file (*BRM_Home*/**sys/cm/taxcodes_map**) as a starting place. See "Calculating Flat Taxes by Using the taxcodes_map File" for information about the file.

2. If you change the name or location of the **taxcodes_map** file, modify the **taxcodes_map** entry in the CM configuration file (*BRM_Home*/**sys/cm/pin.conf**) to point to the new name and location.

3. Modify the source file for the PCM_OP_CUST_POL_TAX_INIT policy opcode (*BRM_SDK_Home*/**source/sys/fm_cust_pol/fm_cust_pol_tax_init.c**) to implement your customizations.

   a. If you plan to load tax code data in a different format than is used by default in the **taxcodes_map** file, define a data structure in **fm_cust_pol_tax_init.c** for your custom tax rates.

   b. Perform any other customizations required in **fm_cust_pol_tax_init.c** to implement your custom tax features.

4. If necessary for your customizations, modify the source file for the PCM_OP_CUST_POL_TAX_CALC policy opcode (*BRM_SDK_Home***/source/sys/fm_cust_pol_tax_calc.c**).

5. Compile **fm_cust_pol_tax_init.c** and **fm_cust_pol_tax_calc.c** by using the makefile in the *BRM_SDK_Home***/source/sys/fm_cust_pol** directory.

6. Copy the compiled shared library files to *BRM_Home***/lib**, replacing the existing versions.

7. Open the CM configuration file (**pin.conf**) and ensure that the following entries in the **fm_module** section are enabled (not commented out):

```
- cm fm_module BRM_Home/lib/fm_cust.a fm_cust_config - pin
- cm fm_module BRM_Home/lib/fm_cust_pol.a fm_cust_pol_config fm_cust_pol_tax_
init pin
```

   Note: The example above shows the entries for AIX systems. The file name extension is **.so** for Solaris, Linux, and HP-UX IA64 systems.

8. Stop and restart the CM.

   See the discussion of starting and stopping the BRM system in *BRM System Administrator's Guide*, for more information.

Retrieving Additional Tax Data from Vertex Communications Tax Q Series

When you perform tax calculation by using Vertex Communications Tax Q Series, it returns tax data in an input flist to the BRM API. The tax data is then processed and stored in the BRM database. For a list of data that Vertex Communications Tax Q Series returns to the BRM API by default, see "Default Tax Data Returned by Vertex Communications Tax Q Series" for more information.

You can also customize BRM to request and store additional data from Vertex Communications Tax Q Series by performing the following:

1. Requesting additional data from Vertex Communications Tax Q Series.

   See "Requesting Additional Data from Vertex Communications Tax Q Series" for more information.

2. Retrieving the additional tax data returned by Vertex Communications Tax Q Series.

   See "Requesting Additional Data from Vertex Communications Tax Q Series" for more information.

3. Storing the additional tax data in the BRM database.

   See "Storing the Requested Vertex Tax Data in the BRM Database" for more information.

Default Tax Data Returned by Vertex Communications Tax Q Series

Vertex Communications Tax Q Series makes its attributes available to other applications through data handles. BRM uses tax data from the following two data handles only:

- The Register Transaction data handle, which defines tax transaction attributes.
- The Register Transaction Tax Detail data handle, which defines tax jurisdiction attributes.

Vertex Communications Tax Q Series sends tax attributes to the BRM API in the following input flist structures:

- **PIN_FLD_TAXES input flist array**: Contains the result of a tax calculation at the transaction level.
- **PIN_FLD_TAXES.PIN_FLD_SUBTOTAL input flist array**: Contains the result of a tax calculation at the jurisdiction level, such as at the Federal level or at the State level. Within each jurisdiction, the taxes are broken into subtypes, such as 911 and DEAF.

Table 4–2 shows the tax attributes that are sent to the BRM API by default and the flist structure in which the attribute is passed:

Table 4–2 Default BRM Tax Attributes and flist Structure

Vertex Communications Tax Q Series Data Handle	Vertex Communications Tax Q Series Attributes	BRM Input Flist Structure
Register Transaction data handle	■ eCtqAttribChargeToLocationMode ■ eCtqAttribChargeToPostalCode ■ eCtqAttribTaxedGeoCodeOverrideCode ■ eCtqAttribInvoiceDate ■ eCtqAttribBilledLines ■ eCtqAttribInvoiceNumber ■ eCtqAttribServiceCode ■ eCtqAttribCategoryCode ■ eCtqAttribCreditCode ■ eCtqAttribDescriptionFlag ■ eCtqAttribTaxedGeoCodeIncorporatedCode ■ eCtqAttribFederalExemptFlag ■ eCtqAttribStateExemptFlag ■ eCtqAttribCountyExemptFlag ■ eCtqAttribCityExemptFlag ■ eCtqAttribWriteJournal **Note:** The eCtqAttribWriteJournal attribute is retrieved from the Register Subsystem data handle.	PIN_FLD_TAXES array
Register Transaction Tax Detail data handle	■ eCtqAttribTrunksTaxed ■ eCtqAttribTaxCode ■ eCtqAttribLinkTaxAmount ■ eCtqAttribRowCount	PIN_FLD_TAXES.PIN_FLD_SUBTOTAL array

For more information about Vertex Communications Tax Q Series data handles and attributes, see the Vertex Communications Tax Q Series documentation.

Requesting Additional Data from Vertex Communications Tax Q Series

You can request additional tax data from Vertex Communications Tax Q Series by customizing the PCM_OP_RATE_POL_PRE_TAX policy opcode. Vertex Communications Tax Q Series returns the requested tax data in the input flist to the PCM_OP_RATE_POL_POST_TAX policy opcode.

To request additional tax data:

- Customize the PCM_OP_RATE_POL_PRE_TAX policy opcode to request data from the following two Vertex data handles: Register Transaction data handle and Register Transaction Tax Detail data handle.
- Add custom output flist fields to the PCM_OP_RATE_POL_PRE_TAX policy opcode. To request additional tax data, add the flist fields shown in bold below to the PIN_FLD_RESULTS output flist array:

```
0 PIN_FLD_RESULTS           ARRAY [0] allocated 20, used 1
1   PIN_FLD_TAXES                 [0] allocated 20, used 1
2     PIN_FLD_FIELD_NAMES     ARRAY [ArrayIndexValue] allocated 20, used 2
3       PIN_FLD_TYPE           ENUM [0] VertexDataType
1   PIN_FLD_SUBTOTAL              [0] allocated 20, used 1
2     PIN_FLD_FIELD_NAMES     ARRAY [ArrayIndexValue] allocated 20, used 2
3       PIN_FLD_TYPE           ENUM [0] VertexDataType
```

  where:

  The PIN_FLD_TAXES structure specifies that you are requesting data from the Vertex Register Transaction data handle. All elements under this array must reference attributes from the Register Transaction data handle only.

  The PIN_FLD_SUBTOTAL structure specifies that you are requesting data from the Vertex Register Transaction Tax Detail data handle. All elements under this array must reference attributes from the Register Transaction Tax Detail data handle only.

  ArrayIndexValue specifies the Vertex Communications Tax Q Series attribute ID that you are requesting. For example, set the PIN_FLD_FIELD_NAMES array index to **304** to request the eCtqAttribTrunksTaxed attribute.

  VertexDataType specifies the Vertex data type ID for the Vertex Communications Tax Q Series attribute you are requesting. Supported Vertex data types include the following:

  - eCtqInt
  - eCtqString
  - eCtqBool
  - eCtqFloat
  - eCtqDouble
  - eCtqLong
  - eCtqDate

For example, the following shows custom output flist fields for requesting additional tax data:

```
0 PIN_FLD_RESULTS                ARRAY [0] allocated 20, used 1
1   PIN_FLD_TAXES                      [0] allocated 20, used 1
2     PIN_FLD_FIELD_NAMES        ARRAY [271] allocated 20, used 2
3       PIN_FLD_TYPE              ENUM [0] 3
1   PIN_FLD_SUBTOTAL                   [0] allocated 20, used 1
2     PIN_FLD_FIELD_NAMES        ARRAY [285] allocated 20, used 2
3       PIN_FLD_TYPE              ENUM [0] 10
```

Retrieving Additional Tax Data

The additional data you request from Vertex Communications Tax Q Series is returned in the input flist of the PCM_OP_RATE_POL_POST_TAX policy opcode. You can customize the BRM API to retrieve the tax data from the input flist fields and process it according to your business needs.

Vertex Communications Tax Q Series returns the additional data in the following PCM_OP_RATE_POL_POST_TAX policy opcode input flist structure under the PIN_FLD_TAXES array or the PIN_FLD_TAXES.PIN_FLD_SUBTOTAL array:

- If passed in the PIN_FLD_TAXES array, the data is for one tax transaction. This information is from the Register Transaction data handle.
- If passed in the PIN_FLD_TAXES.PIN_FLD_SUBTOTAL array, the tax data is for one jurisdiction. This information is from the Register Transaction Tax Detail data handle.

  ```
  1  PIN_FLD_FIELD_NAMES           ARRAY [ArrayIndexValue] allocated 20, used 3
  2    PIN_FLD_TYPE                      ENUM [0] VertexDataType
  2    PIN_FLD_VERTEX_CTQ_BRMDataType   INT [0] BRMTaxData
  ```

  where:

  ArrayIndexValue specifies the Vertex Communications Tax Q Series attribute ID. For example, enter **295** to request data from the eCtqAttribTaxedGeoCode attribute.

  VertexDataType specifies the data type ID of the Vertex Communications Tax Q Series attribute. For example, enter **3** for the eCtqInt data type.

  BRMTaxData is the tax data from the specified Vertex Communications Tax Q Series attribute.

  PIN_FLD_VERTEX_CTQ_*BRMDataType* is the name of the input flist field that contains the tax data. Table 4–3 specifies the name of the BRM input flist field that corresponds to each Vertex data type.

Table 4–3 BRM Input flist for Vertex Data Type

BRM Data Type	Vertex Data Type
INT	eCtqInt
STR	eCtqString
BOOL	eCtqBool
FLOAT	eCtqFloat
DOUBLE	eCtqDouble
LONG	eCtqLong

Table 4–3 (Cont.) BRM Input flist for Vertex Data Type

BRM Data Type	Vertex Data Type
STR **Important:** This field will contain the date in Vertex Communications Tax Q Series format (*CCYYMMDD*). This is different from the mapping for custom input fields, in which **eCtqDate** is mapped to PIN_FLD_VERTEX_CTQ_TIMESTAMP.	eCtqDate

For example, the following shows a simple input flist for the PCM_OP_RATE_POL_POST_TAX policy opcode. The flist fields for the additional tax data are shown in bold.

```
# number of field entries allocated 20, used 2
0 PIN_FLD_POID                 POID [0] 0.0.0.1 /account 27225 0
0 PIN_FLD_EVENT_OBJ            POID [0] 0.0.0.1 /event/billing/cycle/tax 11111 0
0 PIN_FLD_TAXES               ARRAY [0] allocated 20, used 3
1   PIN_FLD_TAXPKG_TYPE         ENUM [0] 0
1   PIN_FLD_TAX              DECIMAL [0] 0.70
1   PIN_FLD_FIELD_NAMES        ARRAY [295] allocated 20, used 3  // Array index =
eCtqAttribTaxedGeoCode
2     PIN_FLD_TYPE              ENUM [0] 10
2     PIN_FLD_VERTEX_CTQ_STR     STR [0] "123456789"
1   PIN_FLD_SUBTOTAL           ARRAY [0] allocated 20, used 10
2     PIN_FLD_TAX              DECIMAL [0] 0.70
2     PIN_FLD_TYPE              ENUM [0] 0
2     PIN_FLD_NAME               STR [0] "US; CA; Sunnyvale; ; 94086"
2     PIN_FLD_AMOUNT_GROSS     DECIMAL [0] 19.88
2     PIN_FLD_PERCENT          DECIMAL [0] 0.035000
2     PIN_FLD_AMOUNT_TAXED     DECIMAL [0] 19.88
2     PIN_FLD_AMOUNT_EXEMPT    DECIMAL [0] 0.00
2     PIN_FLD_SUBTYPE           ENUM [0] 0
2     PIN_FLD_DESCR              STR [0] "Excise"
2     PIN_FLD_LOCATION_MODE     ENUM [0] 0
2     PIN_FLD_FIELD_NAMES       ARRAY [304] allocated 20, used 3  // Array index =
eCtqAttribTrunksTaxed
3       PIN_FLD_TYPE            ENUM [0] 3
3       PIN_FLD_VERTEX_CTQ_INT   INT [0] 1
```

Each PIN_FLD_TAXES array element results in a single balance impact. All tax data from the PIN_FLD_TAXES array element is stored in the BRM **/event** object in the PIN_FLD_BAL_IMPACTS array.

Each PIN_FLD_TAXES.PIN_FLD_SUBTOTAL array element results in a single balance impact for a jurisdiction. All tax data for that jurisdiction is stored in the BRM **/event** object in the PIN_FLD_TAX_JURISDICTIONS array.

Note: In some cases, a Vertex attribute may be missing from the input flist of the PCM_OP_RATE_POL_POST_TAX policy opcode. This could be due to an exception condition. For example:

- If taxes do not apply because of an undefined service code. In this case, the PIN_FLD_SUBTOTAL array will have a zero tax amount, but none of the additional output fields.
- If BRM requested a non-applicable attribute. For example, if BRM requested the eCtqAttribOriginGeoCode attribute when calculating taxes for a wireless service based on the PPU location passed in as a ZIP code.

Storing the Requested Vertex Tax Data in the BRM Database

All additional Vertex Communications Tax Q Series data that you request is returned in the PCM_OP_RATE_POL_POST_TAX policy opcode input flist. To customize the policy opcode to store the additional data in the BRM database:

- Create a custom tax storable class.

  See "Creating a Custom Storable Class for Vertex Tax Data" for more information.

- Customize the PCM_OP_RATE_POL_POST_TAX policy opcode to store the additional tax data in your custom tax storable class.

  See "Storing Vertex Tax Data in Your Custom Storable Class" for more information.

- Configure BRM to *itemize* taxes by jurisdiction.

  See "Itemizing or Summarizing Taxes for Each Jurisdiction Level" for more information.

Creating a Custom Storable Class for Vertex Tax Data

To store the additional data from Vertex Communications Tax Q Series, create a custom tax storable class, such as **/custom/tax/ctq_output**. The structure of the custom tax storable class should be similar to the following flist:

Note: Make sure you define all fields in the custom tax storable class.

```
PIN_FLD_POID
PIN_FLD_ACCOUNT_OBJ
PIN_FLD_EVENT_OBJ
   PIN_FLD_CUSTOM_TAX_BAL_IMPACTS           ARRAY
      PIN_FLD_CUSTOM_TAX_TAXED_GEO_CODE     STR
...
   PIN_FLD_CUSTOM_TAX_TAX_JURISDICTIONS     ARRAY
      PIN_FLD_ELEMENT_ID                    INT
      PIN_FLD_CUSTOM_TAX_TRUNKS_TAXED       INT
```

See "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide* for more information on how to create custom storable classes and custom BRM fields.

Storing Vertex Tax Data in Your Custom Storable Class

To store the additional Vertex Communications Tax Q Series tax data in your custom tax storable class, customize the PCM_OP_RATE_POL_POST_TAX policy opcode as follows:

1. Add the following compiler directives to include Vertex Communications Tax Q Series compiler definitions and copy these files from the Vertex Communications Tax Q Series installation directory to the directory in which the policy source code resides.

   ```
   #include "stda.h"
   #include "ctqa.h"
   ```

2. In the policy source code directory, modify **Makefile** to add the **-DPORT_UNIXANSI Compiler** option.

   For example, you can use the CFLAGS variable as follows:

   ```
   CFLAGS += -DPORT_UNIXANSI
   ```

3. Create a new flist instance of your custom tax storable class and set PIN_FLD_EVENT_OBJ of this instance in the **/event** object.

4. In your custom tax storable class:

   - Add a PIN_FLD_CUSTOM_TAX_BAL_IMPACTS array element for each PIN_FLD_TAXES element passed in the PCM_OP_RATE_POL_POST_TAX input flist.
   - In the PIN_FLD_CUSTOM_TAX_BAL_IMPACTS element, set each custom tax storable class field to its corresponding value from the PIN_FLD_TAXES input flist array. For example, set the PIN_FLD_CUSTOM_TAX_TAXED_GEO_CODE storable class field to the value in the PIN_FLD_VERTEX_CTQ_STR input flist field.
   - Add a PIN_FLD_CUSTOM_TAX_TAX_JURISDICTIONS array element for each PIN_FLD_TAXES.PIN_FLD_SUBTOTAL element passed in the PCM_OP_RATE_POL_POST_TAX input flist.
   - In the PIN_FLD_CUSTOM_TAX_TAX_JURISDICTIONS element, set each custom tax storable class field to its corresponding value from the PIN_FLD_SUBTOTAL input flist array. For example, set the PIN_FLD_CUSTOM_TAX_TRUNKS_TAXED storable class field to the value in the PIN_FLD_VERTEX_CTQ_INT input flist field.
   - Link each PIN_FLD_CUSTOM_TAX_TAX_JURISDICTIONS element to its corresponding parent PIN_FLD_CUSTOM_TAX_BAL_IMPACTS element by using the PIN_FLD_ELEMENT_ID field.

     Note: BRM uses this same method to link PIN_FLD_TAX_JURISDICTIONS elements to corresponding parent PIN_FLD_BAL_IMPACTS elements in the **/event** object.

5. Create an instance of your custom tax storable class in the BRM database by calling the PCM_OP_CREATE_OBJ opcode.

6. Drop the PIN_FLD_FIELD_NAMES arrays from the PCM_OP_RATE_POL_POST_TAX policy opcode output flist, because they are no longer needed.

Note:

- In the rare case that no taxes are computed, the policy input flist will include a single PIN_FLD_TAXES element set to 0 and no PIN_FLD_SUBTOTAL elements. Likewise, when a jurisdiction check is performed to validate the account address, the policy input flist will not include a PIN_FLD_TAXES element. In both cases, you do not need to create a new instance of your custom tax storable class.
- For event-time taxation, the PCM_OP_RATE_POL_POST_TAX policy opcode input flist does not include the PIN_FLD_EVENT_OBJ field.
- For reporting purposes, you can relate a taxed event's default tax fields with the additional tax fields by using the **/event** POID as the join criterion between the **/event** object and your custom tax object.

Modifying Tax Data Before Calculating Taxes

Use the PCM_OP_RATE_POL_PRE_TAX policy opcode to modify data before you send the data to the taxation DM for calculating taxes.

In the default implementation, this policy opcode is called by PCM_OP_RATE_TAX_CALC before determining the tax package to use for tax calculation.

By default, the PCM_OP_RATE_POL_PRE_TAX policy opcode returns the input flist as the output flist. You can customize the policy opcode to perform these functions:

- Identify the geocode for an address and provide it to the taxation DM.
- Add a tax exemption to the input flist before sending the flist to the taxation DM.
- Change the tax exemption from a percentage to a set amount. For example, deduct *x* amount from the taxable amount because the first *x* amount is exempt from taxes.

In addition, the policy opcode source file contains commented code for enhancements available to Vertex Communications Tax Q Series users.

Customizing Vertex Communications Tax Q Series to Override ZIP Codes

For Vertex Communications Tax Q Series, the PCM_OP_RATE_POL_PRE_TAX policy source file contains commented code to instruct the Vertex Communications Tax Q Series taxation software to override the ZIP code. This enables the Vertex DM to charge taxes for non-telephony events by using Vertex Communications Tax Q Series.

To customize Vertex Communications Tax Q Series to override ZIP codes:

1. Open the PCM_OP_RATE_POL_PRE_TAX policy source file with a text editor.
2. Uncomment the code.
3. Define the OVERRIDE_BY_ZIP symbol.
4. Save and close the file.
5. Recompile the policy opcode.

Customizing Vertex Communications Tax Q Series to Provide Custom Input Tax Data

For Communications Tax Q Series, you can modify the PCM_OP_RATE_POL_PRE_TAX policy opcode source code to use a customized input flist to the taxation DM for tax data. The taxation DM checks the input flist. If it includes the PIN_FLD_STATUS_FLAGS field with the value **1**, the DM sets all custom fields included in the flist and then calls the tax package to calculate taxes.

> **Note:** When you create the customized flist, make sure that you express the field values on the basis of Vertex field data types. Add the correct data type and value for each Vertex attribute that you want to customize. If you want to see your customized flist, you can use PIN_ERR_LOG_FLIST in debug mode.

To customize the PCM_OP_RATE_POL_PRE_TAX policy opcode:

1. Open *BRM_SDK_Home*/**source/sys/fm_rate_pol/fm_rate_pol_pre_tax.c** file.
2. Add the code from "Source Code for Customizing PCM_OP_RATE_POL_PRE_TAX" to the **fm_rate_pol_pre_tax()** function after the OVERRIDE_BY_ZIP section.
3. Add optional fields to customize the tax calculation process (the source code includes a comment about how to add new fields).
4. Define the USE_VERTEX_Q_SERIES symbol.
5. Add **stda.h** and **ctqa.h** to the Include section of the file.
6. Set the value of **customized_flag** to **1**, if it is not already set to that value.
7. Save and close the file.
8. Copy the **stda.h** and **ctqa.h** files from the *Vertex_Home*/**inc** directory to the *BRM_SDK_Home*/**source/sys/fm_rate_pol** directory.
9. To the **Makefile**, add CFLAGS += -DPORT_UNIXANSI.
10. Recompile the policy opcode.

Source Code for Customizing PCM_OP_RATE_POL_PRE_TAX

```
#ifdef USE_VERTEX_Q_SERIES
/*****************************************************************************
      /*************************************************************
      * FOLLOWING MANDATORY FIELDS SHOULD BE FILLED WITH CORRECT VALUES.
      * YOU CAN CUSTOMIZE THE MANDATORY AND OPTIONAL FIELDS.
      * HERE IS THE LIST OF MAPPING FIELDS.
      * VERTEX DATATYPE => MAPPING FIELD              =>PORTAL DATATYPE
      *
      * eCtqString    =>PIN_FLD_VERTEX_CTQ_STR        =>PIN_FLDT_STR
      * eCtqInt       =>PIN_FLD_VERTEX_CTQ_INT        =>PIN_FLDT_INT
      * eCtqFloat     =>PIN_FLD_VERTEX_CTQ_FLOAT      =>PIN_FLDT_DECIMAL
      * eCtqDouble    =>PIN_FLD_VERTEX_CTQ_DOUBLE     =>PIN_FLDT_DECIMAL
      * eCtqLong      =>PIN_FLD_VERTEX_CTQ_LONG       =>PIN_FLDT_DECIMAL
      * eCtqDate      =>PIN_FLD_VERTEX_CTQ_DATE       =>PIN_FLDT_TSTAMP
      * eCtqTime      =>PIN_FLD_VERTEX_CTQ_TIME       =>PIN_FLDT_TSTAMP
      * eCtqTimestamp =>PIN_FLD_VERTEX_CTQ_TIMESTAMP  =>PIN_FLDT_TSTAMP
      * eCtqBool      =>PIN_FLD_VERTEX_CTQ_BOOL       =>PIN_FLD_INT
      * eCtqChar      =>PIN_FLD_VERTEX_CTQ_CHAR       =>PIN_FLDT_STR
      * eCtqEnum      =>PIN_FLD_VERTEX_CTQ_ENUM       =>PIN_FLDT_ENUM
      *
      * YOU CAN ADD ONE FIELD LIKE THE FOLLOWING FLIST FORMAT.
```

```
        0 PIN_FLD_TAXES         ARRAY [] allocated 24, used 24
        1 PIN_FLD_FIELD_NAMES   ARRAY [] allocated 20, used 2
                2     PIN_FLD_TYPE             ENUM [0]
                2     PIN_FLD_VERTEX_CTQ_STR    STR [0]
        * USE PROPER MAPPING FIELD TO SET VALUE OF VERTEX FIELD.
        * IF YOU ADD SOME STRING FIELD(eCtqString),THEN SET VALUE FOR THAT FIELD
        * USING PIN_FLD_VERTEX_CTQ_STR.FOR DETAILS,SEE THE ABOVE TABLES.
        *
        *
        * TO OVERRIDE DETAILED TAX INFORMATION (ATTRIBUTES THAT SET IN
        * REGTAX HANDEL IN VERTEX) SUCH AS TAX CODE, TAX TYPE, RATE, AND
        * TAX AUTHROITY, THE FIELD VALUES NEED TO SPECIFY UNDER THE
        * TAX_INPUT ARRAY, WITH INDEX STARTS FROM 1.
        * EX:
        *  0 PIN_FLD_TAXES         ARRAY [] allocated 24, used 24
        *  ....
        *  1     PIN_FLD_TAX_INPUT     ARRAY [1] allocated 20, used 4
        *  2         PIN_FLD_FIELD_NAMES   ARRAY [340] allocated 20, used 2
        *  3             PIN_FLD_TYPE           ENUM [0] 10
        *  3             PIN_FLD_VERTEX_CTQ_STR    STR [0] "40"
        *  2         PIN_FLD_FIELD_NAMES   ARRAY [339] allocated 20, used 2
        *  3             PIN_FLD_TYPE           ENUM [0] 10
        *  3             PIN_FLD_VERTEX_CTQ_STR    STR [0] "0"
        *  2         PIN_FLD_FIELD_NAMES   ARRAY [341] allocated 20, used 2
        *  3             PIN_FLD_TYPE           ENUM [0] 10
        *  3             PIN_FLD_VERTEX_CTQ_STR    STR [0] "X"
        *  2         PIN_FLD_FIELD_NAMES   ARRAY [224] allocated 20, used 2
        *  3             PIN_FLD_TYPE           ENUM [0] 6
        *  3             PIN_FLD_VERTEX_CTQ_DOUBLE DECIMAL [0] 0
        *
        *
        * CUSTOMIZED FIELDS ARE SET INSIDE THE TAXES ARRAY ELEMENTS. WHEN THE
        * CUSTOMIZATION FLAG IS SET, ONLY CUSTOMIZED FIELDS (FIELD_NAMES
        * UNDER TAXES AND FIELD_NAMES UNDER TAX_INPUT) ARE CONSIDERED. REST
        * OF THE FIELDS ARE NEGLECTED.
    ***************************************************************/
        pin_flist_t     *cust_flistp = NULL;
        pin_flist_t     *field_name_flistp = NULL;
        tCtqAttribType  attribute_type = eCtqHandle ;
        time_t          now_t = 0;
        pin_decimal_t   *value_decimal = NULL;
        int32           value_int = 0;
        char            temp_buf[255] = { '\0' };
        pin_cookie_t    cookie = NULL;
        pin_flist_t     *t_flistp = NULL;
        pin_flist_t     *ti_flistp = NULL;
        int32           elemid = 0;

        int32           customized_flag = 1;/* SET TO 1 FOR CUSTOMIZATION */

        /*****************************************************************
         * IMPORTANT:SET customized_flag to 1 IF YOU WANT TO CUSTOMIZE THE
         * VERTEX FIELDS.
         *****************************************************************/
        PIN_FLIST_FLD_SET(cust_flistp, PIN_FLD_STATUS_FLAGS,
                (void *)&customized_flag, ebufp);
        while ((t_flistp = PIN_FLIST_ELEM_GET_NEXT(ret_flistp, PIN_FLD_TAXES,
                &elemid, 1, &cookie, ebufp)) != (pin_flist_t*)NULL) {

                cust_flistp = PIN_FLIST_CREATE(ebufp);
```

```
/* FIELD:eCtqAttribCategoryCode NATURE:Mandatory Field */

field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribCategoryCode, ebufp);
attribute_type = eCtqString;
temp_buf[0]='\0';
strcpy(temp_buf ,"04") ;
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,
        (void *)temp_buf,ebufp);

/* FIELD:eCtqAttribServiceCode NATURE:Mandatory Field */

field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribServiceCode, ebufp);

attribute_type = eCtqString;
temp_buf[0]='\0';
strcpy(temp_buf,"01" );
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,
        (void *)temp_buf,ebufp);

/* FIELD:eCtqAttribOriginGeoCode NATURE:Mandatory Field */

field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribOriginGeoCode, ebufp);
attribute_type = eCtqString;
temp_buf[0]='\0';
strcpy(temp_buf ,"390290740" );
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,
        (void *)temp_buf,ebufp);

/* FIELD:eCtqAttribTerminationGeoCode NATURE:Mandatory Field */

field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribTerminationGeoCode, ebufp);

attribute_type = eCtqString;
temp_buf[0]='\0';
strcpy(temp_buf,"390296350") ;
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,(void *)temp_buf,
        ebufp);

/* FIELD:eCtqAttribInvoiceDate NATURE:Mandatory Field */

field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribInvoiceDate, ebufp);
attribute_type=eCtqDate;
now_t = pin_virtual_time((time_t *)NULL);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_DATE,&now_t,
```

```
        ebufp);

/* FIELD:eCtqAttribTaxableAmount NATURE:Mandatory Field */

field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribTaxableAmount, ebufp);
attribute_type = eCtqDouble;
value_decimal = pbo_decimal_from_str("100.00",ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_DOUBLE,
        (void *)value_decimal,ebufp);


/*********************************************************************
 * YOU CAN ADD OPTIONAL FIELDS LIKE BELOW FLIST
 *
 * field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp,
        PIN_FLD_FIELD_NAMES,eCtqAttribBilledLines, ebufp);
 * attribute_type = eCtqInt;
 * PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
 * int value_int = 01
 * PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_INT,
        (void *)&value_int,ebufp);
 *********************************************************************/
 field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribBilledLines, ebufp);
 attribute_type = eCtqInt;
 value_int = 01;
 PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
 PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_INT,
        (void *)&value_int,ebufp);
field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribSaleResaleCode, ebufp);
attribute_type = eCtqString;
temp_buf[0]='\0';
strcpy(temp_buf ,"S" );
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,
        temp_buf,ebufp);

field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribChargeToGeoCode, ebufp);
attribute_type = eCtqString;
temp_buf[0]='\0';
strcpy(temp_buf ,"390296350" );
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,
        temp_buf,ebufp);

field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribCallMinutes, ebufp);
attribute_type = eCtqDouble;
value_decimal = pbo_decimal_from_str("10.0",ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
```

```
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_DOUBLE,
        (void *)value_decimal,ebufp);

field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribUserArea, ebufp);
attribute_type = eCtqString;
temp_buf[0]='\0';
strcpy(temp_buf ,"Interstate/Toll");
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,temp_buf,
        ebufp);

field_name_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_FIELD_NAMES,
        eCtqAttribInvoiceNumber, ebufp);
attribute_type = eCtqString;
temp_buf[0]='\0';
strcpy(temp_buf ,"DEMO-002");
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
        (void *)&attribute_type,ebufp);
PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,temp_buf,
        ebufp);

  /*********************************************************************
   * YOU CAN OPTIONALLY OVERRIDE DETAILED TAX INFORMATION
   * (ATTRIBUTES THAT SET IN REGTAX HANDEL IN VERTEX) SUCH AS
   * TAX CODE, TAX TYPE, RATE, AND TAX AUTHROITY, THE FIELD
   * VALUES NEED TO SPECIFY UNDER THE TAX_INPUT ARRAY, WITH
   * INDEX STARTS FROM 1.
   *********************************************************************
  /

    ti_flistp = PIN_FLIST_ELEM_ADD(cust_flistp, PIN_FLD_TAX_INPUT, 1,
    ebufp);
    field_name_flistp = PIN_FLIST_ELEM_ADD(ti_flistp, PIN_FLD_FIELD_NAMES,
    eCtqAttribTaxType, ebufp);
    attribute_type = eCtqString;
    temp_buf[0]='\0';
    strcpy(temp_buf,"40") ;
    PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
    (void *)&attribute_type,ebufp);
    PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,(void *)
    temp_buf, ebufp);

    field_name_flistp = PIN_FLIST_ELEM_ADD(ti_flistp, PIN_FLD_FIELD_NAMES,
    eCtqAttribTaxAuthority, ebufp);
    attribute_type = eCtqString;
    temp_buf[0]='\0';
    strcpy(temp_buf,"0") ;
    (field_name_flistp,PIN_FLD_TYPE,
    (void *)&attribute_type,ebufp);
    PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,(void *)
    temp_buf, ebufp);

    field_name_flistp = PIN_FLIST_ELEM_ADD(ti_flistp, PIN_FLD_FIELD_NAMES,
    eCtqAttribTaxCode, ebufp);
    attribute_type = eCtqString;
    temp_buf[0]='\0';
    strcpy(temp_buf,"X") ;
    PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
```

```
            (void *)&attribute_type,ebufp);
            PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_STR,(void *)
            temp_buf, ebufp);

          field_name_flistp = PIN_FLIST_ELEM_ADD(ti_flistp, PIN_FLD_FIELD_NAMES,
          eCtqAttribRate, ebufp);
          attribute_type = eCtqDouble;
          value_decimal = pbo_decimal_from_str("0.0",ebufp);
          PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_TYPE,
          void *)&attribute_type,ebufp);
          PIN_FLIST_FLD_SET(field_name_flistp,PIN_FLD_VERTEX_CTQ_DOUBLE,
          (void *)value_decimal,ebufp);


          PIN_ERR_LOG_FLIST(PIN_ERR_LEVEL_DEBUG,
                  "op_rate_pol_pre_tax customized flist", cust_flistp);
          PIN_FLIST_CONCAT(ret_flistp,cust_flistp,ebufp);
          if (value_decimal) {
                  pbo_decimal_destroy(&value_decimal);
          }

          PIN_FLIST_DESTROY_EX(&cust_flistp,NULL);
*******************************************************************************/
#endif
```

Modifying Tax Data after Calculating Taxes

You can customize the PCM_OP_RATE_POL_POST_TAX policy opcode to modify any data in the output returned by the tax calculation software after tax calculation. For example, you can modify the policy opcode to perform these functions:

- Change how the CM logs any messages that are returned from the external taxation DMs.
- Add a surcharge to the tax amount in the output flist returned from the taxation DM, as shown in the following example:

```
void fm_rate_pol_post_tax(...)
{
/* loop through array of incoming taxes */
  while(t_flistp = PIN_FLIST_ELEM_GET_NEXT(in_flistp, PIN_FLD_TAXES,
      &elemid, 1, &cookie, ebufp)) != (pin_flist_t*)NULL) {
     /* add a surcharge to the SUBTOTAL array */
      cnt = PIN_FLIST_ELEM_COUNT(t_flist, PIN_FLD_SUBTOTAL, ebufp))
      s_flistp = PIN_FLIST_ELEM_ADD(t_flistp, PIN_FLD_SUBTOTAL, cnt++, ebufp);
...
     /* add the entries to the SUBTOTAL array */
      PIN_FLIST_FLD_SET(s_flistp, PIN_FLD_TAX, (void*)&tax, ebufp);
      PIN_FLIST_FLD_SET(s_flistp, PIN_FLD_SUBTYPE, (void*)&taxType, ebufp);
...
  }
}
```

Error Handling: By default, the PCM_OP_RATE_POL_POST_TAX policy opcode returns the input flist as the output flist, without the PIN_FLD_MESSAGES array. It also logs any messages returned from the taxation DM as specified in the **tax_return_loglevel** entry of the CM **pin.conf** file.

The PCM_OP_RATE_POL_POST_TAX policy opcode default implementation handles errors as follows:

- Logs messages as specified in the **tax_return_loglevel** entry in the CM **pin.conf** file. The default is to log only warnings.
- If an error occurs while reading the **tax_return_loglevel** entry, sets the error buffer to INVALID_CONF.

Using Geocodes to Calculate Taxes

A geocode is a geographic code that is used to determine a tax jurisdiction. Different tax calculation software packages use different geocode systems and update them frequently. To use geocodes to calculate taxes, you need to modify the PCM_OP_RATE_POL_PRE_TAX policy opcode to query the geocoder and obtain the geocode for an account during account creation. The geocoder maps the address or postal code of the account to a unique geocode.

If a geocode is available during tax calculation, the tax package uses it to calculate the tax. If a geocode is not available, the tax calculation software chooses the geocode with the highest tax rate from a list of geocodes it finds for that account address.

Note: The tax calculation software itself cannot map an address to a unique geocode.

To use geocodes to calculate taxes:

1. Using Storable Class Editor, create a **/profile** storable class and add the PIN_FLD_GEOCODE field to it.

   See "Creating Custom Fields and Storable Classes" in *BRM Developer's Guide* for more information.

   If you have configured more than one tax calculation package, create an array that contains separate PIN_FLD_GEOCODE fields for the tax calculation package. The index values should represent the tax packages:

   - **0** specifies a custom tax package.
   - **3** specifies the Sales Tax Q Series package.
   - **4** specifies the Communications Tax Q Series package.
2. Modify the PCM_OP_CUST_POL_VALID_NAMEINFO policy opcode to store the geocode in the new PIN_FLD_GEOCODE field of the **/profile** object if the address was validated successfully.
3. Implement the PCM_OP_RATE_POL_PRE_TAX policy opcode to perform the following functions:
   - Obtain the geocode from the **/profile** object.
   - Set the geocode in the PIN_FLD_SHIP_TO or PIN_FLD_SHIP_FROM field in the taxes array in the input flist to the taxation DM.

     For example, the opcode should change this field value:

     ```
     PIN_FLD_SHIP_TO STR [0] "Cupertino; CA; 95012; US; []"
     ```

     to include the geocode in the square brackets []:

     ```
     PIN_FLD_SHIP_TO STR [0] "Cupertino; CA; 95012; US; [050850860]"
     ```

4. Set the location mode to geocode by supplying **1** as the value for the PIN_FLD_LOCATION_MODE field in the PIN_FLD_TAXES array of the input flist.

   When you set the location mode to geocode, the taxation DM passes the geocode instead of the address to the tax calculation module.

   For example:

   ```
   PIN_FLD_LOCATION_MODE  ENUM[0] 1
   ```

After calculating the taxes, the taxation DM returns the jurisdiction (address and geocode) where tax rates were applied in the PIN_FLD_TAXES array of the output flist:

```
PIN_FLD_NAME STR[0] "US: CA: Santa Clara: Cupertino: 95012 : [050850860]"
```

Adding Tax Information to Accounts

To add tax information to an account, use the PCM_OP_CUST_SET_TAXINFO opcode. This opcode adds or updates values for the following fields in the **taxinfo** array of an **/account** object:

- PIN_FLD_VAT_CERT
- PIN_FLD_EXEMPTIONS
- PIN_FLD_INCORPORATED_FLAG
- PIN_FLD_RESIDENCE_FLAG

For example, when an account is created and a VAT certificate number or exemption information is provided, PCM_OP_CUST_SET_TAXINFO is called. PCM_OP_CUST_SET_TAXINFO performs these functions:

- Modifies existing data or adds information to the PIN_FLD_TAXINFO array of the **/account** object by using the field values provided in the input flist.
- If a VAT certificate number is not provided, replaces the value of PIN_FLD_VAT_CERT in the account object with an empty ("") string.
- If exemption information is not provided, deletes the PIN_FLD_EXEMPTIONS array.
- If the field values provided in the input flist contain the PIN_FLD_INCORPORATED_FLAG or PIN_FLD_RESIDENCE_FLAG flag, adds or updates these values in the account object.

Validating Tax Information

To customize how tax information is validated, use the PCM_OP_CUST_POL_VALID_TAXINFO policy opcode. This policy opcode validates the VAT certificate number to prevent nonvalid numbers, which cause failures in tax calculations.

By default, the PCM_OP_CUST_POL_VALID_TAXINFO policy opcode checks for the VAT certificate data and performs these functions:

- If the VAT certificate number provided is a string, zero in length, returns the validation result **PASS**.
- If the following **tax_valid** entry in the CM **pin.conf** file is not set or is set to **0**, returns the validation result **FAIL**.

  ```
  #Enables or disables validation of the VAT certificate number.
  #1 means Enable; 0, which is the default, means Disable.
  ```

```
- fm_cust_pol  tax_valid 0
```

- If the **tax_valid** entry in the CM **pin.conf** file is set to **1**, invokes the PCM_OP_RATE_TAX_CALC opcode to validate the VAT certificate number by passing the special command flag PIN_FLD_COMMAND set to PIN_CUST_TAX_VAL_VAT_CERT. If you have implemented custom tax calculation, you will need to validate the VAT certificate number only if the PIN_FLD_COMMAND input field is set to PIN_CUST_TAX_VAL_VAT_CERT.

For more information, see "About the PREP and VALID Opcodes" in *BRM Developer's Guide*.

For example, instead of calling the PCM_OP_RATE_TAX_CALC opcode, you can query your custom VAT validation function by rewriting the function **do_vat_cert_validation** in the PCM_OP_CUST_POL_VALID_TAXINFO policy opcode source file.

5

Calculating Taxes Utilities

This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) calculating taxes utilities.

load_tax_supplier

Use the **load_tax_supplier** utility to load one or more tax suppliers into the BRM database's **/config/tax_supplier** object. You define the tax suppliers in either the *BRM_Home*/**setup/scripts/pin_tax_supplier.xml** file or another file that uses the same format. The format of the XML file is specified in the **pin_tax_supplier.xsd** schema file in the *BRM_Home*/**xsd** directory.

You can define only one tax supplier as the default tax supplier for your database.

Caution: The **load_tax_supplier** utility overwrites existing tax suppliers. If you are updating tax suppliers, you cannot load new tax suppliers only. You must load complete sets of tax suppliers each time you run the **load_tax_supplier** utility.

Important:

- To connect to the BRM database, the **load_tax_supplier** utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator's Guide* for more information.
- Each time you run this utility, you must restart the Connection Manager (CM).

Note: You cannot load separate **/config/tax_supplier** objects for each brand. All brands use the same object.

See "About Tax Suppliers" for information about tax suppliers.

Location

BRM_Home/**bin**

Syntax

```
load_tax_supplier [-d | -v | -t | -h] pin_tax_supplier.xml
```

Parameters

-d
Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors, but the data has not been loaded into the database.

-v
Displays information about successful or failed processing as the utility runs.

To redirect the output to a log file, use the following command. Replace *filename*.**log** with the name of the log file:

load_tax_supplier *pin_tax_supplier*.**xml–v >** *filename*.**log**

Note: If a file with the same name exists, it is overwritten.

-t
Checks the validity of the XML file, but doesn't create an object.

-h
Displays the syntax and parameters for this utility.

pin_tax_supplier
- The name and location of the file that defines the tax suppliers. The default **pin_tax_supplier.xml** file is in *BRM_Home*/**setup/scripts**.
- If you do not run the utility from the directory in which the file is located, you must include the complete path to the file, for example:

  load_pin_tax_supplier *BRM_Home*/**setup/scripts/pin_tax_supplier.xml**

Results

If the **load_tax_supplier** utility doesn't notify you that it was successful, look in the **default.pinlog** file to find any errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.