



AvaTax for Communications

Telecom User Manual

Release: 9.19.1801.1
Document: TM_00101_0049
Date: 12/21/2017

Avalara for Communications - Contact Information	
Address	Avalara, Inc. 513 South Mangum Street, Suite 100 Durham, North Carolina, 27701
Toll Free	800-525-8175
Corporate Web Site	http://communications.avalara.com/
Comms Platform Site	https://communications.avalara.net
Email	communicationsupport@avalara.com

Document Revision History

The Revision History log lists the date and description of the most recent revisions or versions of the document.

Date	Version	Description
03/25/2016	0028	Avalara branding updates to reflect the transition to the new company and product names have been incorporated into this document. Please see Appendix H – Avalara Product Names for specific changes in product references and descriptions.
04/25/2016	0029	Updated with table of Tax Categories in Section 2.4 ; new tax type 457 in Section 4.3.5 Tax Types . Additional release number and version updates throughout Section 8 .
05/25/2016	0030	New tax type 458 in Section 4.3.5 Tax Types ; updated support site screenshots in Section 8.4.1 Download the Monthly update . Reformatted Section 7.8 API Listing table. Additional release number and version updates throughout Section 8 .
06/24/2016	0031	Added new tax types 459 – 462 in Section 4.3.5 Tax Types ; 9 new transaction/service pairs in Section 2.1 Transaction and Service Types ; Updates in Section 5.5.7 Customsort.exe to command line and arguments; Release number, version and header file updates throughout Section 8 .
07/25/2016	0032	Added new tax type 463 in Section 4.3.5 Tax Types ; Release number, version and header file updates throughout Section 8 .
08/25/2016	0033	Release number and version updates on cover and in Section 8.2
09/23/2016	0034	Added new tax types, 464 – 468 in Section 4.3.5 Tax Types . Added new transaction/service pair 14/15 in Section 2.2 . Added new country, Senegal, in Section 2.3.9 . Release number, version and header file updates on the cover and throughout Section 8 .
10/25/2016	0035	Updated descriptions in Section 3.1.4.1 Federal Exempt and Section 3.2 Incorp. Release number, version and header file updates on the cover and throughout Section 8 .
11/23/2016	0036	Added new tax type 469 in Section 4.3.5 Tax Types . All references to ‘reverse’ tax calculations in the AFC engine have been updated and renamed to reflect the current naming convention which is ‘tax inclusive’ calculations. As a result, 16 APIs in Section 7.8.1 API Listings have been updated and renamed. Added safe harbor APIs in Section 7.8.107 EZTaxClearSafeHarborOverride and Section 7.8.108 EZtaxSetSafeHarborTAMOverride . Release number, version and header file updates on the cover and throughout Section 8 .
12/22/2016	0037	Added new tax types 470 and 473-475 in Section 4.3.5 Tax Types . Updates in Section 3.6.2 Determining the Taxing Jurisdiction for Cellular and Section 3.6.3 Determining the Taxing Jurisdiction for VoIP . Added new Section 3.6.4 Determining the Taxing Jurisdiction within Canada . Release number, version and header file updates on the cover and throughout Section 8 .
01/27/2017	0038	Release number, version and header file updates on the cover and throughout Section 8 . Also updated copyright year.
02/24/2017	0039	Removed references to the Locations field throughout document as well as transaction/service pairs 7/23 and 7/88 in Section 2.2 Valid Transaction/Service Pairings . Release number, version and header file updates on the cover and throughout Section 8 .
03/24/2017	0040	Updated table under Section 7.8 API Listings , updated Section 7.8.78 EZTaxPrivateLine and added Section 7.8.79 EZTAXPrivateLineAdj . Release

Date	Version	Description
		number, version and header file updates on the cover and throughout Section 8 .
04/25/2017	0041	Added new transaction/service pair 10/655 in Section 2.2 Valid Transaction/Service Pairings . Added new tax type 469 in Section 4.3.5 Tax Types . Updates to Section 4.7 Exclusions , Section 6.3 Federal or State Exclusion and Section 7.8.97 EZtaxSetStateExclusions . Release number, version and header file updates on the cover and throughout Section 8 .
05/25/2017	0042	Release number, version and header file updates on the cover and throughout Section 8 .
06/27/2017	0043	Added Section 3.5.7 Support for India . Added 12 new tax types, 66 and 477 – 487 in Section 4.3.5 Tax Types . All references to ‘Customer Mode’ in the AFC engine have been updated and renamed to reflect the current naming convention which is ‘Invoice Mode.’ As a result, 2 APIs in Section 7.8 API Listings have been updated and renamed. Added two new APIs, Section 7.8.65 EZTaxJTypeEx and Section 7.8.89 EZTaxPTTypeEx . Release number, version and header file updates on the cover and throughout Section 8 .
07/27/2017	0044	Added note related to timestamp/invoice date passed in Section 6.2.4.1 Invoice Date . Removed deprecated API, EZTaxJType and added 6 new prorated adjustment APIs in Section 7.8 API Listings . Release number, version and header file updates on the cover and throughout Section 8 .
08/25/2017	0045	Updated tax type 482 (name only). Release number, version and header file updates on the cover and throughout Section 8 .
09/25/2017	0046	Updated name and description for tax type 220, added 2 new tax types 488 and 489 in Section 4.3.5 Tax Types . Added new APIs, EZTaxGetCountry ID and EZTaxGetStateID in Section 7.8 API Listings . Release number, version and header file updates on the cover and throughout Section 8 .
10/25/2017	0047	Added new tax type 490 in Section 4.3.5 Tax Types . Updated references to support site web site address in Section 8.4 Monthly Update Procedures . Updated Avalara contact information (address and support site). Removed Appendix H – Avalara Product Names . Release number, version and header file updates on the cover and throughout Section 8 .
11/22/2017	0048	Added new tax type 491 and 492 in Section 4.3.5 Tax Types . Release number, version and header file updates on the cover and throughout Section 8 .
12/21/2017	0049	Added 6 new countries to the table in Section 2.3.9 Telephony Transaction and Service Types . Release number, version and header file updates on the cover and throughout Section 8 .

Table of Contents

1.	Introduction.....	1
1.1	Installation	1
1.2	Getting Started.....	1
1.3	The AFC Manual	1
1.3.1	Mapping	1
1.3.2	Transactions	2
1.3.3	VPN Transaction and Service Types.....	2
1.3.4	Tax Calculations.....	2
1.3.5	Utilities	2
1.3.6	The Advantage of Using Functions	2
1.3.7	General Product Information	4
2.	Mapping.....	6
2.1	Transaction and Service Types	6
2.2	Valid Transaction/Service Pairings	6
2.2.1	File transervdescsau.txt.....	29
2.3	Transaction Mapping Guidelines.....	29
2.3.1	Long Distance Transaction and Service Types	29
2.3.2	Wireless Transaction and Service Types.....	31
2.3.3	Enhanced Service Transaction and Service Types	31
2.3.4	Internet Transaction and Service Types	32
2.3.5	Local Service Transaction and Service Types	32
2.3.6	Sales Transaction and Service Types	35
2.3.7	Shipping Transaction and Service Types.....	36
2.3.8	Natural Gas Transaction and Service Types.....	36
2.3.9	Telephony Transaction and Service Types.....	36
2.3.10	Cable Television Transaction and Service Types.....	39
2.3.11	Satellite Television Transaction and Service Types.....	39
2.3.12	Voice over Internet Protocol (VoIP) Transaction and Service Types.....	40
2.3.13	Payphone Transaction and Service Types.....	44

2.3.14	Software Transaction and Service Types	44
2.3.15	Timesharing Transaction and Service Types.....	44
2.3.16	Satellite Phone Transaction and Service Types.....	45
2.3.17	VPN Transaction and Service Types.....	45
2.4	Tax Categories.....	45
3.	Transactions.....	46
3.1	Customer Information.....	47
3.1.1	Business Class Indicator.....	48
3.1.2	Customer Number (Primary Output Key)	48
3.1.3	Customer Type	48
3.1.4	Exemption Levels.....	48
3.2	Incorp	49
3.2.1	Invoice Number	49
3.2.2	JCode Exemption Levels	49
3.2.3	Lifeline Flag.....	50
3.2.4	Service Class Indicator	50
3.2.5	Service Level Number	50
3.2.6	Tax Type Exemptions	50
3.2.7	Optional Fields.....	51
3.3	Company Information	53
3.3.1	Company Identifier.....	53
3.3.2	Facilities Based Flag	53
3.3.3	Franchise Flag.....	53
3.3.4	Regulated / Unregulated Flag	53
3.4	Transaction Data	54
3.4.1	Charge	54
3.4.2	Date	54
3.4.3	Lines	55
3.4.4	Minutes	55
3.4.5	Sale Type	55
3.4.6	Service Type	55

3.4.7	Transaction Type	55
3.5	Taxing Jurisdiction Identification Information.....	56
3.5.1	The JCode Jurisdiction Identification	56
3.5.2	NPANXX	59
3.5.3	Zip Codes	59
3.5.4	Zip Plus 4	60
3.5.5	Canadian Postal Codes	60
3.5.6	FIPS Codes	60
3.5.7	Support for India	62
3.6	Jurisdiction Identification Details	63
3.6.1	Determining the Taxing Jurisdiction for Wireline	63
3.6.2	Determining the Taxing Jurisdiction for Cellular.....	67
3.6.3	Determining the Taxing Jurisdiction for VoIP	67
3.6.4	Determining the Taxing Jurisdiction within Canada	67
4.	AFC Calculations.....	68
4.1	Meeting the Requirements of Specific Tax Issues	68
4.1.1	Tax Type Exemptions.....	69
4.1.2	Tax Adjustments	70
4.1.3	Tax Overrides.....	71
4.1.4	Tax Rate Brackets	77
4.1.5	Surcharges.....	78
4.1.6	Prorated Taxes.....	78
4.1.7	Discount Adjustment.....	78
4.1.8	Tier at Transactions	81
4.1.9	Tax on Tax Until no Effect.....	82
4.1.10	Historical Tax Rates	82
4.1.11	Taxed Taxes	82
4.2	Tax Logging	83
4.3	Returned Taxes	83
4.3.1	Taxes Table	84
4.3.2	Returned Tax Information	84

4.3.3	Returned Tax Information Using Invoice Mode.....	84
4.3.4	Tax Grouping	86
4.3.5	Tax Types.....	88
4.3.6	Nexus.....	107
4.3.7	Exclusions	108
5.	Utilities.....	109
5.1	Utility Selection	110
5.1.1	Reporting Utilities.....	110
5.1.2	File Management Utilities	111
5.2	Specifying a Log File at Run Time	111
5.3	Log.sum File	112
5.4	General Tips When Using Utilities	112
5.5	AFC Utilities.....	113
5.5.1	asciilog.exe	115
5.5.2	batchsplit.exe	116
5.5.3	commerge.exe.....	117
5.5.4	comptnum.exe	118
5.5.5	comrpt.exe	120
5.5.6	csf20.exe	121
5.5.7	customsort.exe.....	122
5.5.8	ezcomprep.exe	123
5.5.9	ezlog_ns.exe	125
5.5.10	ezlogcust.exe	127
5.5.11	ezlogcustios.exe	129
5.5.12	ezlogcustpts.exe	131
5.5.13	ezlogcustptslnl.exe	133
5.5.14	EZTax_20.exe.....	135
5.5.15	EZTaxappend.exe.....	135
5.5.16	EZTaxappendf.exe	136
5.5.17	log no tax transactions	137
5.5.18	srtcdf20.exe.....	140

5.5.19	srtcdf20p.exe.....	142
5.5.20	srtcomcust20.exe	144
5.5.21	srtcomma20.exe.....	145
5.5.22	srtcomma20l.exe	148
5.5.23	srtcomma20ld.exe.....	151
5.5.24	srtcommadetail.exe.....	154
5.5.25	srtrev20l.exe.....	156
5.5.26	upsize_log.exe	161
6.	BATCH FILE PROCESSING.....	162
6.1	Batch Processing Description	162
6.2	AFC CDS Input File Specifications	164
6.2.1	Taxing Jurisdiction Identification Information	165
6.2.2	Customer Information	167
6.2.3	Company Information	170
6.2.4	Transaction Information.....	171
6.3	Federal or State Exclusion	172
6.4	Accumulating the Log.....	173
6.5	EZTax_20 Utility Specification	174
6.6	Deprecated CDF20 File Structure.....	175
7.	The C/C++ API	177
7.1	Language Interfaces for AFC	177
7.2	Configuration	178
7.2.1	Filelocs.txt File	178
7.2.2	Filelocs.sta File.....	179
7.3	General Overview of AFC API Integration	180
7.4	Detailed Discussion of AFC API Integration	181
7.4.1	Tax Adjustments	181
7.4.2	Obtaining Jurisdiction and Address Information From JCodes and PCodes	181
7.4.3	Obtaining Jurisdiction and Address Information From JCodes and PCodes	181
7.4.4	Obtaining Tax Information From Transactions	182
7.5	Preparing the AFC Application Programmer Interface (API) Interface	184

7.6	AFC API Function Calls	184
7.6.1	Retrieving Taxes	185
7.6.2	Multi-Threading.....	187
7.6.3	Multiple Sessions.....	188
7.6.4	Session Management	189
7.6.5	Session Pooling.....	190
7.7	Sample Coding	191
7.7.1	Using EZTaxGetRates to Build an Override.....	191
7.8	API Listings	192
7.8.1	EZTaxAdjDebitJEx.....	200
7.8.2	EZTaxAdjDebitNEx	201
7.8.3	EZTaxAdjDebitPEx.....	202
7.8.4	EZTaxAdjDebitTaxInclusiveJCode (EZTaxAdjDebitRevJCode deprecated).....	203
7.8.5	EZTaxAdjDebitTaxInclusiveNPAN (EZTaxAdjDebitRevNPAN deprecated).....	204
7.8.6	EZTaxAdjDebitTaxInclusivePCode (EZTaxAdjDebitRevPCode deprecated)	205
7.8.7	EZTaxAdjDebitTaxInclusiveZip (EZTaxAdjDebitRevZip deprecated)	206
7.8.8	EZTaxAdjDebitZEx.....	208
7.8.9	EZTaxAdjJCodeEx.....	209
7.8.10	EZTaxAdjNPANEx	210
7.8.11	EZTaxAdjPCodeEx	211
7.8.12	EZTaxAdjProRateJCode (EZTaxAdjProRateJCodeEx deprecated)	212
7.8.13	EZTaxAdjProRateNPAN (EZTaxAdjProRateNPANEx deprecated)	213
7.8.14	EZTaxAdjProRatePCode (EZTaxAdjProRatePCodeEx deprecated)	214
7.8.15	EZTaxAdjProRateThisJCode (EZTaxAdjProRateThisJCodeEx deprecated)	215
7.8.16	EZTaxAdjProRateThisPCode (EZTaxAdjProRateThisPCodeEx deprecated).....	217
7.8.17	EZTaxAdjProRateZip (EZTaxAdjProRateZipEx deprecated)	219
7.8.18	EZTaxAdjTaxInclusiveJCode (EZTaxAdjRevJCode deprecated)	220
7.8.19	EZTaxAdjTaxInclusiveNPAN (EZTaxAdjRevNPAN deprecated)	221
7.8.20	EZTaxAdjTaxInclusivePCode (EZTaxAdjRevPCode deprecated)	222
7.8.21	EZTaxAdjTaxInclusiveZip (EZTaxAdjRevZip deprecated)	223
7.8.22	EZTaxAdjTPPEX	225

7.8.23	EZTaxAdjZipEx.....	227
7.8.24	EZTaxCalcJurisdiction.....	228
7.8.25	EZTaxClearExclusion	229
7.8.26	EZTaxClearTSR	230
7.8.27	EZTaxClose.....	231
7.8.28	EZTaxCountryToPCode	232
7.8.29	EZTaxDbVersion.....	233
7.8.30	EZTaxDebitJEx.....	234
7.8.31	EZTaxDebitNEx	235
7.8.32	EZTaxDebitPEx	236
7.8.33	EZTaxDebitTaxInclusiveJCode (EZTaxDebitRevJCode deprecated)	237
7.8.34	EZTaxDebitTaxInclusiveNPAN (EZTaxDebitRevNPAN deprecated)	239
7.8.35	EZTaxDebitTaxInclusivePCode (EZTaxDebitRevPCode deprecated)	240
7.8.36	EZTaxDebitTaxInclusiveZip (EZTaxDebitRevZip deprecated)	241
7.8.37	EZTaxDllVersion	243
7.8.38	EZTaxExit	244
7.8.39	EZTaxExitSessionEx.....	245
7.8.40	EZTaxFlushToLogEx.....	246
7.8.41	EZTaxFreeRates	247
7.8.42	EZTaxFtoPCodeEx	248
7.8.43	EZTaxGetAddressEx	249
7.8.44	EZTaxGetCustomLog.....	250
7.8.45	EZTaxGetCountryID	251
7.8.46	EZTaxGetCustomLogCount.....	252
7.8.47	EZTaxGetHandle	253
7.8.48	EZTaxGetJurisdiction	254
7.8.49	EZTaxGetLogName	255
7.8.50	EZTaxGetLogV914.....	256
7.8.51	EZTaxGetLogV914Count.....	257
7.8.52	EZTaxGetRates.....	258
7.8.53	EZTaxGetSession.....	259

7.8.54	EZTaxGetStateID.....	260
7.8.55	EZTaxGetTaxCatV98	261
7.8.56	EZTaxGetTaxDescription.....	262
7.8.57	EZTaxGetTSR.....	263
7.8.58	EZTaxGroupResults.....	264
7.8.59	EZTaxInitDirEx.....	265
7.8.60	EZTaxInitEx	266
7.8.61	EZTaxInitExMT	267
7.8.62	EZTaxInitV914.....	268
7.8.63	EZTaxInitV98.....	269
7.8.64	EZTaxJCodeEx	270
7.8.65	EZTaxJtoPCodeEx.....	271
7.8.66	EZTaxJTypeEx (EZTaxJType deprecated).....	272
7.8.67	EZTaxJurisdiction	273
7.8.68	EZTaxMaxTaxCount	274
7.8.69	EZTaxNextAddressEx	275
7.8.70	EZTaxNextCustomerEx.....	276
7.8.71	EZTaxNPANEx	277
7.8.72	EZTaxNtoJCodeEx	278
7.8.73	EZTaxNTTypeEx	279
7.8.74	EZTaxOldOvrJCode	280
7.8.75	EZTaxOvrJCodeEx	281
7.8.76	EZTaxOvrNPANEx.....	282
7.8.77	EZTaxOvrPCodeEx.....	283
7.8.78	EZTaxOvrZipEx	284
7.8.79	EZTaxPCodeEx	285
7.8.80	EZTaxPrivateLine	286
7.8.81	EZTaxPrivateLineAdj	287
7.8.82	EZTaxProRateJCodeEx	288
7.8.83	EZTaxProRateNPANEx.....	289
7.8.84	EZTaxProRatePCodeEx.....	290

7.8.85	EZTaxProRateThisJCodeEx.....	291
7.8.86	EZTaxProRateThisPCodeEx	292
7.8.87	EZTaxProRateZipEx	293
7.8.88	EZTaxPtoFipsEx.....	294
7.8.89	EZTaxPtoJCodeEx.....	295
7.8.90	EZTaxPTTypeEx	296
7.8.91	EZTaxRestoreEx	297
7.8.92	EZTaxTaxInclusiveJCode (EZTaxRevJCode deprecated)	298
7.8.93	EZTaxTaxInclusiveNPAN (EZTaxRevNPAN deprecated).....	299
7.8.94	EZTaxTaxInclusivePCode (EZTaxRevPCode deprecated).....	300
7.8.95	EZTaxTaxInclusiveZip (EZTaxRevZip deprecated).....	301
7.8.96	EZTaxSessionDbVersion.....	303
7.8.97	EZTaxSetInvoiceModeEx.....	304
7.8.98	EZTaxSetInvoiceModeV98	305
7.8.99	EZTaxSetNexus	306
7.8.100	EZTaxSetStateExclusion	307
7.8.101	EZTaxSetStateNexus	308
7.8.102	EZTaxSetWorkingDir	309
7.8.103	EZTaxThisJCodeEx.....	310
7.8.104	EZTaxThisPCodeEx	311
7.8.105	EZTaxTPPEx.....	312
7.8.106	EZTaxWriteToLogEx	313
7.8.107	EZTaxWriteToLogV914	314
7.8.108	EZTaxZipEx.....	315
7.8.109	EZTaxZtoJCodeEx.....	316
7.8.110	EZTaxZtoPCodeEx	317
7.8.111	EZTaxClearSafeHarborOverride	318
7.8.112	EZTaxSetSafeHarborTAMOverride.....	319
8.	Appendices A - G.....	320
8.1	Appendix A EZTaxStruct.h	321
8.2	Appendix B EZTaxDefine.h	335

8.3	Appendix C EZTaxProto.h	384
8.4	Appendix G Monthly Update Procedure	389
8.4.1	Download the Monthly Update	389
8.4.2	Important Files in Update	392
8.4.3	Monthly Maintenance	392
8.5	Help Guide	394
8.5.1	Troubleshooting	394
8.5.2	FAQ's	395

1. Introduction

1.1 Installation

Refer to the **AvaTax for Communications (AFC) Installation Manual** for complete instructions regarding the installation of AFC on your specific platform.

1.2 Getting Started

There are several AFC integration methods available, each allowing for its use based on the client's needs.

- AFC SaaS Standard: This method is provided for lower transaction-count customers, allowing for the use of AFC to obtain tax calculations without having to maintain the program on their systems.
- On-Site Options:
 - AFC's API interface: provides the user complete control over access to tax calculations and the data returned, along with the flexibility supported by the report utilities.
- AFC SaaS Pro: AFC SaaS Pro is an XML web service used to calculate taxes. Applications send a single transaction to the AFC SaaS web service over the Internet. The taxes are calculated and immediately returned to the client application. Users have the capability of sending and receiving transactions 24 hours a day, 7 days a week. This service is ideal for adding the ability to tax purchases made from online stores and AFC SaaS Pro customers are provided with reports for tax compliance filing.

1.3 The AFC Manual

The AFC Manual has been configured to provide an easy and logical progression for learning AFC and its features, as well as a quick reference (fully linked on the electronic version of the manual). From the initial step of mapping to the final creation of reports based on the processed information, the conceptual simplicity of AFC is fully explained.

1.3.1 Mapping

Prior to performing tax calculations, client transactions and services must be matched, or "mapped," to the Valid Transaction and Service Pairings defined in AFC. Refer to Section 2 Mapping for the full explanation of how this is accomplished.

1.3.2 Transactions

Once the transactions and services have been mapped, transaction records can be passed to AFC. Section 3 Transactions explains what AFC expects for each part (or field) contained in a transaction record and provides the definitions and reasons for selecting one of the valid options to use.

1.3.3 VPN Transaction and Service Types

Refer to Table 2-14 for a list of combined transaction and service types that are used with VPN transaction types.

1.3.4 Tax Calculations

The AFC Engine accepts the transaction record(s) and performs the tax calculations based upon the information within the record.

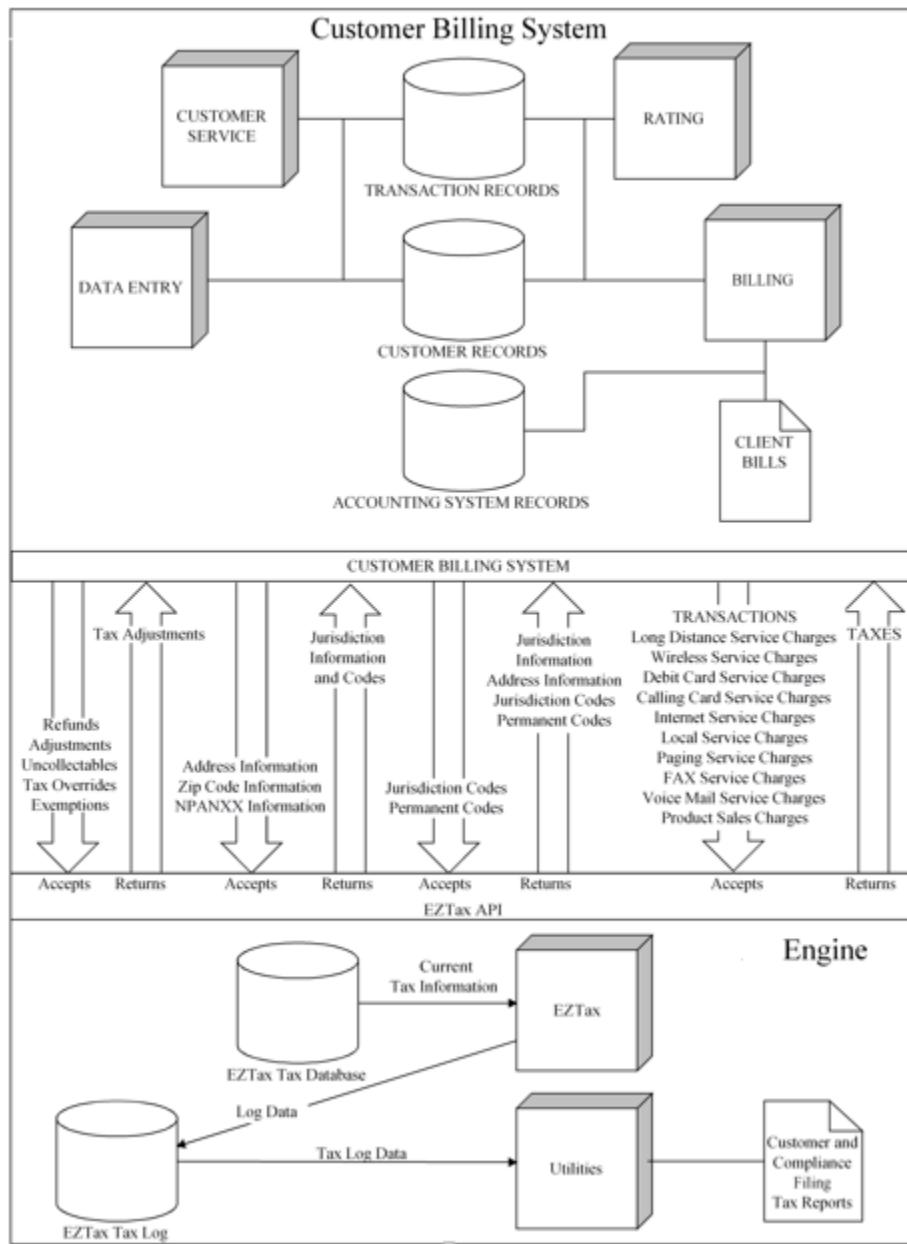
1.3.5 Utilities

Once the taxation has been performed, Utilities can be used to provide the tax information in selected formats that meet specific requirements and perform file management functions. The many options available are detailed in the Utilities section of this document

1.3.6 The Advantage of Using Functions

Section 7 provides specific details of the expanded capabilities provided to users that integrate AFC to the billing system using functions. Refer to the figure shown below. The billing system passes information to AFC, as calls are being rated for billing. AFC calculates all required taxes and returns the tax information to the billing system per transaction. In addition, AFC stores all tax data generated in its databases and provides reports that facilitate tax filing and provide insight to the rating, billing and taxing processes.

Figure 1-1 AFC Operation Diagram



Adjustment information is returned to the billing system and is utilized to update tax data for report generation and compliance filing. AFC also provides facilities that allow users to insert tax overrides and exempt a transaction from taxes at federal, state, county and local authority levels. AFC also provides the capability to limit exemptions to a specific jurisdiction and to exempt specific taxes.

Using the AFC API is the most efficient method available for interfacing with the AFC system. Due to the superior performance of AFC, there is very little performance difference between a system running without taxation and one running with taxation generation by AFC.

1.3.7 General Product Information

Telecom Types

- Business Classifications – Allows for specification of business as CLEC or ILEC.
- Service Classifications – Allows for specification of business as Primary Long Distance or Local Primary.
- Franchise – Allows for the indication that company provides services sold pursuant to a franchise agreement between the carrier and jurisdiction.
- Facilities – Allows for specification of the carrier delivering the service as facilities based.

Customer Attributes

- Company Identifier – Allows the AFC user to indicate a specific company that may be included in their customer record and outputted to the tax log to identify company tax log records.
- Lifeline – Allows for a customer to be designated as a Lifeline participant.

Enhanced Flexibility for Users

User reporting and organizational activities have been enhanced with the addition of user defined fields. This empowers the user to include additional user specific information within transaction records, thereby enhancing reports and data files once processed by the AFC Engine. Note that the Batch Processing utility takes full advantage of the additional transaction fields.

Enhanced Capabilities for Processing Specialized Taxes

AFC maintains tax rate histories and applies them according to the transaction date. This allows for tax calculations to process transactions that require dated tax information.

- Billable Flag – Allows for the identification of a compliance only tax or fee that is used for filing and not passed on to the user. It is noted as a non-billable item in the tax log.
- Compliance Flag – Allows for the identification of a tax or fee that is a tax compliant item and that the tax or fee should be sent to the log. The reporting utilities can identify and generate compliance and billing reports. All existing taxes currently in AFC are compliance taxes or fees. A charge with the compliance set to false in logic is billable but the charge would not be placed into the log for compliance or reported in compliance reports. This flag provides EZdata users with the ability to pass non-tax charges to the customer through AFC.

- Rate at Final - Allows for a specialized application of tax brackets. Each bracket specifies the tax rate for the “Total Tax” range that it spans. The culmination of all of the brackets results in a tax rate “look-up” table for total transaction amounts.
- Tier At Transaction - This logic flag indicates that tax will be calculated using graduated tax brackets on a transaction basis instead of a customer invoice basis.
- Tax On Tax Until No Effect - Configuration settings in EZTax.cfg determine one of two methods to calculate Tax on Tax. If the configuration setting is left unchanged, AFC will perform a single iteration of the tax on tax calculation.

Additional Functions to Support New Capabilities

An expanded library of functions has been created, the greatest impact of which falls on the Adjustment and Pro-Rated activities.

- Adjustment Functions – Additional functions are available to allow for adjustments of debit and pro-rated transactions.
- Pro-Rated Allowable Functions – Allow for a pro-rated percentage to be used to calculate the taxable amount on rate tax and calculate the percentage of the line and fixed amounts. AFC identifies taxes that are not allowed to be assessed on a percentage basis internally, and the entire tax amount is calculated.
- The Get Rates function has been added to retrieve all possible taxes when given a location by PCode. The structure of the returned tax information is compatible with the new override structure, so that specific changes can be applied to the existing tax data to create override entries.

New Version Tracking

A version number has been embedded into the AFC database to ensure database compatibility between the AFC software and the AFC database. The AFC Engine compares version numbers on session initialization. If incompatible version numbers are detected, a message is written to the status file and AFC system will not return a valid session handle.

Additional Tax Types

AFC now supports over 450 Tax Types to allow for highly specific tax type designation.

Increased Accuracy

The Tax Amount and Time calculations have been modified to increase accuracy and further reduce rounding errors.

2. Mapping

The AFC software provides an extensive selection of Transaction and Service Types to meet your taxation requirements with a single taxation package. The correct matching, or “mapping,” of your company products and services to the Valid Transaction and Service Pairings defined in AFC is an essential process that must be completed accurately for taxation to be calculated and applied properly to all transactions. If an incorrect pairing is passed to the AFC engine then no tax is usually returned.

Avalara support is available to assist should any questions arise in determining the correct Transaction and Service pair to use.

2.1 Transaction and Service Types

AFC stipulates a unique pair of numbers for each Transaction and Service Type. The first number defines the Transaction Type and the second number defines the Service Type. Transaction Types and Service Types are combined (or “paired”) to uniquely describe a Valid Transaction / Service Pair for a transaction.

2.2 Valid Transaction/Service Pairings

Table 2-1 provides a complete listing of identifiers used to define valid Transaction / Service Pairings.

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
0	0	No Tax/No Tax	Usage of this mapping combination will ensure that no federal, state or local taxes are returned.
1	1	Interstate/Toll	Interstate toll calls, MRC (monthly recurring charges), and other related service-type charges and features. (Call originates and terminates in different states.)
1	2	Interstate/Toll-Free	Interstate and international toll-free calls, MRCs, and other related service type charges and features. (Same as Interstate Toll call above, however the owner of the toll-free number pays all the applicable long distance charges.)
1	3	Interstate/WATS	Interstate and international WATS service charge, MRC, and other related service type charges and features. (Same as Interstate Toll call above, but often sold as a set-pricing scheme for a designated long distance calling area.)
1	4	Interstate/Private Line	Interstate or international private line service charge, MRC, and other related service type charges and features. (Charges are for a service in which the service originates at the customer's premises and connects only to a designated termination location. No switching or access to other third party lines. The private line will cross over a state or country border.)

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
1	12	Interstate/International Toll	International calls that originate in the US, terminate outside the US, and are billed to a US address.
1	14	Interstate/Late Charge	Late charge imposed on customers of interstate or international LD services.
1	16	Interstate/900	Interstate or international 900 service charge, MRC and other related service type charges and features. (Same as Interstate Toll call above, but the caller of the 900 number pays for the applicable long distance charges.)
1	27	Interstate/Data	Interstate or international data service charge, MRC, and other related service type charges and features. (This combination is appropriate for transmissions that carry data exclusively. Use Private Line if any component is voice. This is a recommended mapping for data transmissions carried over DSL, ATM, T-1, frame relay lines and other non-voice services. This is not for Internet Access provided over DSL lines.)
1	54	Interstate/Directory Assistance	Charges for Directory Assistance calls that cross state boundaries.
1	562	Interstate/Local Loop	Local loop charge that is not part of a local exchange service and is sold in conjunction with an interstate private line.
1	635	Interstate/ Toll Free Number	Monthly recurring charge for access to a toll free number. This represents the interstate portion.
2	1	Intrastate/Toll	Intrastate toll call, MRCs, and other related service types charges and features. (Call originates and terminates within the same state.)
2	2	Intrastate/Toll-Free	Intrastate toll-free calls, MRC, and other related service type charges and features. (Same as Intrastate Toll call above, however the owner of the toll-free number pays all the applicable long distance charges.)
2	3	Intrastate/WATS	Intrastate WATS service charge, MRC, and other related service type charges and features. (Same as Intrastate Toll call above, but often sold as a set-pricing scheme for a designated long distance calling area.)
2	4	Intrastate/Private Line	Intrastate private line service charge, MRC, and other related service type charges and features. (Charges are for a service in which the service originates at the customer's premises and connects only to a designated termination location. No switching or access to other third party lines. The private line will be entirely within a state and not cross a state border.)
2	5	Intrastate/Local Exchange	The basic flat rate for intrastate local exchange service. This transaction/service types will include any applicable long distance access line charges, or other local service-related fees and charges. (Does not include local feature charges.)
2	14	Intrastate/Late Charge	Late charge imposed on customers of intrastate LD services.

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
2	16	Intrastate/900	Intrastate 900 service charge, MRC and other related service type charges and features. (Same as Intrastate Toll call above, but the caller of the 900 number pays for the applicable long distance charges.)
2	27	Intrastate/Data	Intrastate data service charge, MRC, and other related service type charges and features. (This combination is appropriate for transmissions that carry data exclusively. Use Private Line if any component is voice. This is a recommended mapping for data transmissions carried over DSL, ATM, T-1, frame relay lines and other non-voice services. This is not for Internet Access provided over DSL lines.)
2	54	Intrastate/Directory Assistance	Charges for Directory Assistance calls that are contained wholly in one state.
2	630	Intrastate/Private Line (10% Rule)	Intrastate private line service charge, MRC, and other related service type charges and features. (Charges are for a service in which the service originates at the customer's premises and connects only to a designated termination location. The private line will be entirely within a state and not cross a state border.) Traffic on this type of line is considered mixed use and interstate traffic exceeds 10 percent; thus the revenues are treated as interstate for Universal Service contribution purposes.
2	631	Intrastate/Data (10% Rule)	Intrastate data service charge, MRC, and other related service type charges and features. (This combination is appropriate for transmissions that carry data exclusively. Use Private Line if any component is voice. This is a recommended mapping for data transmissions carried over DSL, ATM, T-1, frame relay lines and other non-voice services. This is not for Internet Access provided over DSL lines.) Traffic on this type of line is considered mixed use and interstate traffic exceeds 10 percent; thus the revenues are treated as interstate for Universal Service contribution purposes.
2	635	Intrastate/Toll Free Number	Monthly recurring charge for access to a toll free number. This represents the intrastate portion.
3	6	Other/Access Charge	Access for a service that is not already defined as a transaction/service type. Catch-all local access charge category.
		Other/Local Loop	Local loop charge that is not part of a local exchange service and does not fall within any other transaction/service type category. (Local exchange providers who are billing local loop charges for local exchange services should map this charge under transaction/service type 7-5 [Local Local Exchange].)
3	9	Other/Directory Ads	Directory advertisement charges.
3	14	Other/Late Charge	Category for late charges that were originally taxed using one of the "Other" (3) transaction categories.

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
3	34	Other/Conference Bridge	Charges for connecting conference participants utilizing a conferencing bridge. This transaction should be used when transactions are interstate or cannot be separated from intrastate services.
3	37	Other/Equipment Rental	Charges for Rental of Equipment.
3	38	Other/Wire Maintenance Plan	Monthly recurring charges for inside wiring maintenance between customer phone and the local carrier's demarc box.
3	46	Other/PICC	This combination should be used for the charge assessed by either a LEC or LD company for maintaining record of an end user's choice of LD carrier.
3	47	Other/No Pick PICC	This combination should be used for the charge assessed by a LEC for maintaining record that end users choose not to declare an LD carrier. This combination will only return the necessary State Taxes. It is important to distinguish this transaction from 3/46 because No Pick PICC's are not subject to FUSF.
3	57	Other/Data Processing	Charge for the manipulation of user's data. This is not to be confused with the transmission of data.
3	96	Other/No Pick PICC Bundle	This combination should be used for the charge assessed by a LEC for maintaining record that end users choose not to declare an LD carrier. This combination will only return the necessary State Taxes. It is important to distinguish this transaction from 3/97 because No Pick PICC's are not subject to FUSF. This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return FET.
3	97	Other/PICC Bundle	This combination should be used for the charge assessed by either a LEC or LD company for maintaining record of an end users choice of LD carrier. It will return the proper Federal taxes, such as USF, in addition to necessary State Taxes, such as Sales Tax. This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return FET
3	570	Other/Directory Listing	Directory listing charges.
3	575	Other/Conference Bridge-Intrastate	Charges for connecting intrastate conference participants utilizing a conferencing bridge. This transaction should be used when intrastate services do not include dial in or dial out services.
3	576	Other/Conference Bridge-Intrastate w Dial In	Per-minute and per-participant charges for dial-in service provided in conjunction with connecting conference participants utilizing a conferencing bridge when all participants are located within one state.
3	589	Other/Conference Bridge-Interstate	Charges for connecting interstate conference participants utilizing a conferencing bridge.
3	593	Other/Info Svcs-Private Physical Trans	A service providing information to private customers by physical means such as paper.
3	594	Other/Info Svcs-Private Electronic Trans	A service providing information to private customers by electronic means.

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
3	597	Other/Info Svcs-Public-Electronic Trans	A service providing the passive receipt of information other than financial account or securities trading data to the public by electronic means.
3	598	Other/Info Svcs-Public Physical Trans	A service providing information to the public by physical means such as paper.
3	599	Other/E-mail Hosting Service	A service providing e-mail hosting to customers.
3	600	Other/Real Property Rental	Rental of Real Property Space
3	602	Other/Services-Professional	A service rendered for a fee in one of the learned professions.
3	603	Other/Online services	Access to a computer through a remote terminal that allows retrieval of stored data created by the service provider.
3	608	Other/Conference Bridge Interstate w Dial-In	Per-minute and per-participant charges for dial-in service provided in conjunction with connecting conference participants utilizing a conferencing bridge when all participants are located in different states.
3	614	Other/Telecom Equipment Rental	Charge for renting telecommunications equipment.
3	632	Other/Service Contracts	An optional contract to cover repairs including parts and labor. This type of service contract is sold at the same time the product being covered by the service contract is sold.
3	638	Other/Security Monitoring Services	A fee paid for the service of monitoring the security of real or tangible personal property.
3	639	Other/Streaming Internet Video	The purchase of video via the internet. The purchaser does not retain possession of the video.
3	644	Other/Info Svcs-Pub Elec Trans (Fin & Securities)	A service providing the passive receipt of financial account or securities trading data on to the public by electronic means.
4	7	Non-Recurring/Service	One-time charge for the actual provisioning of manual service to a phone system or account. All manual repair services should fall into this category. (This designation should not be used for administrative fees or service change fees.)
4	8	Non-Recurring/Install	One-time charge for the installation, administration, modification, or termination of a telecommunication service account or service. (Use for install fee, change order fees, add-service fees, and termination account/service fees. Not for repair/service fees.)
4	11	Non-Recurring/Activation	One-time charge for the activation of a local exchange service account. (Local only.)
4	14	Non-Recurring/Late Charge	Category for late charges that were originally taxed using one of the "Non-Recurring" (4) transaction categories.

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
5	6	Internet/Access Charge	Charges for internet access services, MRC, and other related service type charges and features. (Both per-minute and flat fee amounts for internet access services will be mapped to this transaction/service type.)
5	7	Internet/Service	One-time charge for manual, physical service to an internet system or account, such as a truck roll to the customer premise. All manual repair services should fall into this category. This designation should not be used for administrative fees or service change fees.
5	8	Internet/Install	One-time charge for the installation, administration, modification, or termination of an internet service account or service. This should be used for install fee, change order fees, add-service fees, and termination account/service fees, but not for repair/service fees.)
5	10	Internet/Usage	Used for internet access sales that are not a monthly recurring charge, but are sold to customers on a per usage basis.
5	11	Internet/Activation	One-time charge for the activation of an internet service account. It is mutually exclusive of the other internet charges.
5	29	Internet/Web Hosting	Charges for internet web hosting, MRC, and other related service type charges and features.
5	58	Internet/Access Line	A telecommunications line purchased, used, or sold by a provider of Internet access to provide Internet access as long as the charges are distinguishable from other uses.
6	6	Paging/Access Charge	Basic monthly flat-rate charges for paging services. (Mutually exclusive of the other paging charges.)
6	10	Paging/Usage	Paging charges for usage. Charges are in addition to any services covered in the monthly access charge. (Mutually exclusive of the other paging charges.)
6	11	Paging/Activation	One-time charges for activating a paging account. (Mutually exclusive of the other paging charges.)
6	13	Paging/Equipment Repair	Charges for paging equipment repair and service. (Mutually exclusive of the other paging charges.)
7	4	Local/Private Line	Local private line service charge, MRC, and other related service type charges and features. (Charges are for a service in which the service originates at the customer's premises and connects only to a designated termination location. No switching or access to other third party lines. The private line will originate and terminate entirely within a single city. If the private line is charged on a basis of time and distance per call or used to make tolls calls outside of the local calling areas for a set periodic/flat fee charge, then the private line should be mapped as a toll, toll free, or WATS type of service.)

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
7	5	Local/Local Exchange Services	Charges for monthly recurring charge, usage, local loop, local flat rates and other similar charges for local telecom service. This will also include dial tone and any applicable long distance access line charges, or other local service-related fees and charges. (Does not include local feature charges. See below)
7	14	Local/Late Charge	Category for late charges that were originally taxed using one of the Local (7) transaction categories.
7	20	Local/FCC Subscriber Line Fee	Charge for recovering the cost of connecting the customer premise to the local phone network, often referred to as the Federal Subscriber Line Charge (SLC) or the End User Common Line (EUCL) Charge. The FCC has established per-line caps for primary and secondary line charges. This charge is not a tax or a governmental assessment, but an authorized charge for the recovery of cost for providing local exchange service.
		Local/Number Portability Recovery	Fixed, monthly charge through which local telephone companies may recover some of the costs associated with providing local number portability service. This is not the LNP Administrative Fee.
7	21	Local/Lines	Designates the number of lines a local service customer is using. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of lines designated in the lines field.)
7	24	Local/PBX/Trunk	Designates the number of PBX trunks a local service customer is using. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of lines designated in the lines field. Used in conjunction with 7/41 and 7/566.)
7	27	Local/Data	Local data service charge, MRC, and other related service type charges and features. (This combination is appropriate for transmissions that carry data exclusively. Use Private Line if any component is voice. This is a recommended mapping for data transmissions carried over DSL, ATM, T-1, frame relay lines and other non-voice services. This is not for Internet Access provided over DSL lines.)
7	30	Local/Local Feature Charges	Charges and fee for additional feature charges of local exchange services. (Includes services such as call waiting, caller ID, call blocking, call forwarding, etc.)
7	40	Local/Centrex / DID Extension	Designates the number of Centrex / Direct Inward Dialing extensions a local service customer is using. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of extensions designated in the lines field. Used in conjunction with 7/42 and 7/587.)

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
7	41	Local/PBX Extension	Designates the number of PBX extensions a local service customer is using. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of extensions designated in the lines field. Used in conjunction with 7/24 and 7/566.)
7	42	Local/Centrex Trunk	Designates the number of Centrex trunks a local service customer is using. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of trunks designated in the lines field. Used in conjunction with 7/40 and 7/587.)
7	43	Local/Invoice	Mapping category for transactions on a per invoice basis. (Tax is based per invoice per account per billing cycle. Taxable amount or numbers of lines are irrelevant for this transaction/service type.)
7	45	Local/High Capacity Trunk	Designates the number of High Capacity Trunks a customer is using. High Capacity Trunks are usually defined as T1 or greater. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of High Capacity trunks designated in the lines field. Used in conjunction with 7/580 and 7/582.)
7	84	Local/Late Charge Bundle	Category for late charges that were originally taxed using one of the "Local" (7) transaction categories associated with bundled transactions. This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return FET.
7	85	Local/Local Exchange Bundle	Charges for monthly recurring charge, usage, local loop, local flat rates and other similar charges for local telecom service. This will also include dial tone and any applicable long distance access line charges, or other local service-related fees and charges. (Does not include local feature charges. See below). This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return FET.
7	86	Local/FCC Subscriber Line Fee Bundle	Charge for recovering the cost of connecting the customer premise to the local phone network, often referred to as the Federal Subscriber Line Charge (SLC) or the End User Common Line (EUCL) Charge. The FCC has established per-line caps for primary and secondary line charges. This charge is not a tax or a governmental assessment, but an authorized charge for the recovery of cost for providing local exchange service. This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return Federal Excise Tax (FET).

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
		Local/Number Portability Recovery Bundle	Fixed, monthly charge through which local telephone companies may recover some of the costs associated with providing local number portability service. This is not the LNP Administrative Fee. This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return FET.
7	87	Local/Lines Bundle	Designates the number of lines a local service customer is using. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of lines designated in the lines field.). This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return FET.
7	89	Local/PBX Trunk Bundle	Designates the number of PBX trunks a local service customer is using. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of lines designated in the lines field. Used in conjunction with 7/92 and 7/567.) This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return FET.
7	90	Local/Local Feature Charge Bundle	Charges and fee for additional feature charges of local exchange services. (Includes services such as call waiting, caller ID, call blocking, call forwarding, etc.) This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return FET.
7	91	Local/Centrex Extension Bundle	Designates the number of Centrex / Direct Inward Dialing extensions a local service customer is using. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of extensions designated in the lines field. Used in conjunction with 7/93 and 7/588.) This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return FET.
7	92	Local/PBX Extension Bundle	Designates the number of PBX extensions a local service customer is using. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of extensions designated in the lines field. Used in conjunction with 7/89 and 7/567.) This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return FET.
7	93	Local/Centrex Trunk Bundle	Designates the number of Centrex trunks a local service customer is using. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of trunks designated in the lines field. Used in conjunction with 7/91 and 7/588.) This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return FET.

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
7	94	Local/Invoice Bundle	Mapping category for transactions on a per invoice basis. (Tax is based per invoice per account per billing cycle. Taxable amount or numbers of lines are irrelevant for this transaction/service type.) This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return FET.
7	95	Local/High Capacity Trunk Bundle	Designates the number of High Capacity Trunks a customer is using. High Capacity Trunks are usually defined as T1 or greater. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of High Capacity trunks designated in the lines field. Used in conjunction with 7/581 and 7/583.)
7	566	Local/PBX Outbound Channel	Designates the number of voice grade communications channels leaving a subscriber's premises through a PBX connecting the subscriber's premises to the public switched network. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of extensions designated in the lines field. Used in conjunction with 7/24 and 7/41.)
7	567	Local/PBX Outbound Channel Bundle	Designates the number of voice grade communications channels leaving a subscriber's premises through a PBX connecting the subscriber's premises to the public switched network. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of extensions designated in the lines field. Used in conjunction with 7/89 and 7/92.) This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return FET.
7	580	Local/High Capacity Extension	Designates the number of extensions a local customer is using on a High Capacity Trunk. High Capacity Trunks are usually defined as T1 or greater. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of High Capacity Extensions designated in the lines field. Used in Conjunction with 7/45 and 7/582.)
7	581	Local/High Capacity Extension Bundle	Designates the number of extensions a local service customer is using on a High Capacity Trunk. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of extensions designated in the lines field. Used in conjunction with 7/95 and 7/583.) This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return FET.

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
7	582	Local/High Capacity Outbound Channel	Designates the number of outbound channels a local service customer is using on a High Capacity Trunk. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of outbound channels designated in the lines field. Used in conjunction with 7/45 and 7/580.)
7	583	Local/High Capacity Outbound Channel Bundle	Designates the number of outbound channels a local service customer is using on a High Capacity Trunk. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of extensions designated in the lines field. Used in conjunction with 7/95 and 7/581.) This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return FET.
7	587	Local/Centrex Outbound Channel	Designates the number of outbound channels a local service customer is using on a Centrex Trunk. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of outbound channels designated in the lines field. Used in conjunction with 7/40 and 7/42)
7	588	Local/Centrex Outbound Channel Bundle	Designates the number of outbound channels a local service customer is using on a Centrex Trunk. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of extensions designated in the lines field. Used in conjunction with 7/91 and 7/93) This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return FET.
7	612	Local/FCC Subscriber Line Fee Multi Line	Charge for recovering the cost of connecting a multi-line customer premise to the local phone network, often referred to as the Federal Subscriber Line Charge (SLC) or the End User Common Line (EUCL) Charge. The FCC has established per-line caps for primary and secondary line charges. This charge is not a tax or a governmental assessment, but an authorized charge for the recovery of cost for providing local exchange for multi-line service.
7	613	Local/FCC Subscriber Line Fee Multi Line Bundle	Charge for recovering the cost of connecting a multi-line customer premise to the local phone network, often referred to as the Federal Subscriber Line Charge (SLC) or the End User Common Line (EUCL) Charge. The FCC has established per-line caps for primary and secondary line charges. This charge is not a tax or a governmental assessment, but an authorized charge for the recovery of cost for providing local exchange multi-line service. This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return Federal Excise Tax (FET).

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
7	623	Local/Centrex Invoice	Mapping category for transactions on a per invoice basis. (Tax is based on per invoice per account per billing cycle. Taxable amount or numbers of lines are irrelevant for this transaction/service type.)
7	625	Local/Customer Premise Equip Rental	Rental of equipment located at a subscriber's premises that enable customers to access local communications services as defined by the IRS as it will return FET.
7	641	Local/FCC Subscriber Line Charge Centrex	Charge for recovering the cost of connecting the Centrex customer premise to the local phone network, often referred to as the Federal Subscriber Line Charge (SLC) or the End User Common Line (EUCL) Charge. The FCC has established per-line caps for primary and secondary line charges. This charge is not a tax or a governmental assessment, but an authorized charge for the recovery of cost for providing local exchange service. This transaction is intended only for taxation of the Subscriber Line Fee (not the actual fees for Centrex services or the line counts on Centrex systems).
7	642	Local/FCC Subscriber Line Charge Centrex Bundle	Charge for recovering the cost of connecting the Centrex customer premise to the local phone network, often referred to as the Federal Subscriber Line Charge (SLC) or the End User Common Line (EUCL) Charge. The FCC has established per-line caps for primary and secondary line charges. This charge is not a tax or a governmental assessment, but an authorized charge for the recovery of cost for providing local exchange service. This transaction is intended only for taxation of the Subscriber Line Fee (not the actual fees for Centrex services or the line counts on Centrex systems). This service type should be used when passing bundled local service as defined in IRS Notice 2006-50, as it will not return Federal Excise Tax (FET).
8	10	Fax/Usage	Charges for fax services, MRCs, or other related service-type fee and charges.
9	6	Voice Mail/Access Charge	Basic monthly flat-rate charges for voice mail services.
9	10	Voice Mail/Usage	Voice mail charges for usage. Charges are in addition to any services provided in the monthly access charge.
9	11	Voice Mail/Activation	One-time charge for activating a voice mail account.
9	14	Voice Mail/Late Charge	Category for late charges that were originally taxed using one of the Voice Mail (9) transaction categories..
10	15	Sales/Product	General sales tax rates.
10	31	Sales/Use	General use tax rates.
10	32	Sales/Debit	Calculation of sales tax on a debit charge (prepaid charge) that is determined by state law to be a point of sale transaction.
10	63	Sales/Restocking Fee – Rental	Fee charged to reimburse the cost of restocking a rented item. The returned item cannot be modified in any form.

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
10	64	Sales/Restocking Fee – Purchase	Fee charged to reimburse the cost of restocking a purchased item. The returned item cannot be modified in any form.
10	65	Sales/Partial Credit	A credit that is less than the full amount of the original purchase. The reason for the credit reduction must be due to a restocking or handling type fee.
10	103	Sales/Sales Tax and FUSF	This transaction/service pair returns Sales Tax and FUSF. Avalara does not recommend using this combination for any type of telecom service. However, if you feel you provide a service that is only subject to Sales Tax and FUSF, then you can use this transaction/service combination.
10	565	Sales/Debit-Wireless	The Point-of-Sale (POS) purchase of prepaid, pay-as-you-go wireless services sold by the phone carrier or a party controlled by the phone carrier.
10	568	Sales/Central Office Equipment-Sales	Sale of tangible property to a telecommunications provider for the provision of phone service.
10	569	Sales/Central Office Equipment-Use	Use of tangible property to a telecommunications provider for the provision of phone service.
10	643	Sales/Debit-Wireless (Indirect Non-Carrier Sale)	The Point-of-Sale (POS) purchase of prepaid, pay-as-you-go wireless services sold by a party other than the phone carrier or a party controlled by the phone carrier.
10	655	Sales/Locked Cell Phone	Purchase of cell phone equipment restricted to a particular cell phone network by a locking code.
11	17	Shipping/FOB Origin	Shipping charge for FOB origin transactions. (Shipping charges only.)
11	18	Shipping/FOB Destination	Shipping charge for FOB destination transactions. (Shipping charges only.)
12	6	Natural Gas/Access Charge	Basic flat-rate charges for natural gas services. (Non-telecom. Natural gas only.)
12	19	Natural Gas/Consumption	Natural gas charges for consumption of natural gas. (Non-telecom. Natural gas only.)
13	6	Cellular/Access Charge	Basic monthly flat-rate charge for cellular/wireless service.
13	10	Cellular/Usage	Cellular/wireless per-minute and/or per-use charges. Charges are in addition to any monthly access or roaming charges billed to customer.
13	11	Cellular/Activation	One-time charge for activating a cellular/wireless account.
13	14	Cellular/Late Charge	Category for late charges that were originally taxed using one of the Cellular (13) transaction categories.
13	30	Cellular/Feature Charge	Charges and fees for additional feature charges of Cellular services. (Includes services such as call waiting, caller ID, call blocking, call forwarding, etc.)
13	33	Cellular/Roaming Charge	Per-use, per-minute charges for cellular use outside of the designated service area of the providing carrier.

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
13	43	Cellular/Invoice	Mapping category for transactions on a per invoice basis. (Tax is based per invoice per account per billing cycle. Taxable amount or numbers of lines are irrelevant for this transaction/service type.)
13	49	Cellular/Interstate Usage	For use when carrier is passing actual traffic and not using safe harbor percentages. Cellular/wireless per-minute and/or per-use interstate charges. Charges are in addition to any interstate monthly access or interstate roaming charges billed to customer.
13	50	Cellular/Intrastate Usage	For use when carrier is passing actual traffic and not using safe harbor percentages. Cellular/wireless per-minute and/or per-use intrastate charges. Charges are in addition to any intrastate monthly access or intrastate roaming charges billed to customer.
13	51	Cellular/International Usage	Portion of MRC, or per-minute charges, attributable to calls that originate inside the United States and terminate outside the United States.
13	98	Cellular/Access Number	For use when carrier is passing actual traffic and not using safe harbor percentages or for passing transactions for multiple line accounts. Designates the number of access numbers assigned to an account.
13	99	Cellular/Interstate Access Charge	For use when carrier is passing actual traffic and not using safe harbor percentages. Designates the portion of the basic monthly access charge that is interstate.
13	100	Cellular/Intrastate Access Charge	For use when carrier is passing actual traffic and not using safe harbor percentages. Designates the portion of the basic monthly access charge that is intrastate.
13	101	Cellular/Interstate Roaming	For use when carrier is passing actual traffic and not using safe harbor percentages. Per-use, per-minute charges for interstate cellular use outside of the designated service area of the providing carrier.
13	102	Cellular/Intrastate Roaming	For use when carrier is passing actual traffic and not using safe harbor percentages. Per-use, per-minute charges for intrastate cellular use outside of the designated service area of the providing carrier.
13	572	Cellular/Digital Download	The purchase of goods such as ringtones downloaded to a cell phone.
13	577	Cellular/Enhanced Features	Charges and fees for additional feature charges of wireless services which are separate from voice transmission related features as defined by the FCC. (Includes services such as voicemail, interactive voice response, audiotext information services, and protocol processing.)
13	591	Cellular/Access Charge-No Contract	Basic monthly flat rate charge for cellular/wireless service that is sold without a contract.

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
13	592	Cellular/Access Number-Not Contract	For use when carrier is passing actual traffic and not using safe harbor percentages. Designates the number of access numbers assigned to a wireless account that is sold without a contract.
13	610	Cellular/Access Early Termination Fees	A fee charged for early termination of Cellular service.
13	622	Cellular/Text Message	A fee charged to cellular customers for Text Messaging services
14	15	International/Product	Supply of goods for consideration within countries other than Canada, USA, and US territories.
14	25	International/USA Inbound	International calls inbound to the USA that are billed to an international address. Outbound international calls should be mapped as Interstate/International Toll calls [1/12]. (Call must be billed to a non-USA address.)
14	658	International/Product - India Interstate Supply	Supply of goods for consideration within India in which the location of the supplier and the place of supply occur in different states or territories.
14	659	International/Product - India Intrastate Supply	Supply of goods for consideration within India in which the location of the supplier and the place of supply occur in the same state or territory.
15	7	Telephony/Service	All telephone service in countries other than the USA, Canada, Puerto Rico and the Virgin Islands.
15	624	Telephony/Wireless Service	All wireless telephone service in countries other than the USA, Canada, Puerto Rico and the Virgin Islands.
15	627	Telephony/Internet Access	All Internet Access in countries other than Canada, USA and US territories.
15	629	Telephony/Messaging Services	All Messaging Services in countries other than Canada, USA and US territories.
15	656	Telephony/Service - India Interstate Supply	Telephone service in India in which the location of the supplier and the place of supply occur in different states or territories.
15	657	Telephony/Service - India Intrastate Supply	Telephone service in India in which the location of the supplier and the place of supply occur in the same state or territory.
16	6	Cable Television/Access Charge (Alias Basic Service)	Basic monthly flat-rate charge for cable television service.
16	8	Cable Television/Install	One-time charge for the installation of any cable television service.
16	13	Cable Television/Equipment Repair	Charges for cable equipment repair and service.
16	14	Cable Television/Late Charge	Category for late charges that were originally taxed using one of the Cable (16) transaction categories.
16	35	Cable Television/Premium Service	Premium monthly flat-rate charge for cable television premium channel(s) service.
16	36	Cable Television/Pay Per View Service	Pay per view monthly charges for cable television pay per view service.
16	37	Cable Television/Equipment Rental	Equipment (box/switch) monthly charges for cable television.

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
16	39	Cable Television/TV Guide	Charge for TV Guide Sourcing publication for cable television services.
16	584	Cable Television/Digital Channel Tier	Charge for cable television digital tier service.
16	610	Cable Television Early Termination Fees	A fee charged for early termination of Cable Television Service.
16	615	Cable Television/Equipment Sales	Sales of cable television equipment.
16	654	Cable Television/ Equipment Rental Basic	Equipment (box/switch) monthly charges for cable television that provide basic service only.
18	6	Satellite Television/Access Charge (Alias Basic Service)	Basic monthly flat-rate charge for satellite television service.
18	8	Satellite Television/Install	One-time charge for the installation of any satellite television service.
18	13	Satellite Television/Equipment Repair	Charges for satellite equipment repair and service.
18	14	Satellite Television/Late Charge	Category for late charges that were originally taxed using one of the Satellite (18) transaction categories.
18	35	Satellite Television/Premium Service	Premium monthly flat rate charge for satellite television premium channel(s) service.
18	36	Satellite Television/Pay Per View Service	Pay per view monthly charges for satellite television pay per view service.
18	37	Satellite Television/Equipment Rental	Equipment (box/switch) monthly charges for satellite television.
18	39	Satellite Television/TV Guide	Charge for TV Guide Sourcing publication for satellite television service.
19	6	VoIP/Access Charge	Basic monthly flat rate for VoIP service.
19	8	VoIP/Install	Charge for installation of VoIP services.
19	11	VoIP/Activation	One-time charges for activating a VoIP account. (Mutually exclusive of the other VoIP charges.)
19	13	VoIP/Equipment Repair	Charge for repair of equipment necessary to make VoIP calls.
19	14	VoIP/Late Charge	Category for late charges that were originally taxed using one of the VoIP (19) transaction categories.
19	21	VoIP/Lines	Designates the quantity of numbers a VoIP customer is using. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of lines designated in the lines field.) This service type will return E911 at the landline rate regardless of whether it is paired with the VoIP or VoIPA transaction type.
19	30	VoIP/Feature Charge	Charges and fees for additional feature charges of VoIP services. (Includes services such as call waiting, caller ID, call blocking, call forwarding, etc.)
19	37	VoIP/Equipment Rental	Charge for renting equipment necessary to make VoIP phone calls.

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
19	41	VoIP/PBX Extension	Designates the number of VoIP PBX extensions a VoIP service customer is using. Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of extensions designated in the lines field. Used in conjunction with 19/566 and 19/578.
19	43	VoIP/Invoice	Mapping category for transactions on a per invoice basis. (Tax is based per invoice per account per billing cycle. Taxable amount or numbers of lines are irrelevant for this transaction/service type.)
19	48	VoIP/Wireless Access Charge	This will tax similar to Cellular/Access Charge. Federal and State USF are applied, but at the wireless safe harbor rate.
19	49	VoIP/Interstate Usage	Portion of MRC, or per-minute charges, attributable to calls that cross state lines but do not leave the United States.
19	50	VoIP/Intrastate Usage	Portion of MRC, or per-minute charges, attributable to calls that do not cross state lines.
19	51	VoIP/International Usage	Portion of MRC, or per-minute charges, attributable to calls that originate inside the United States and terminate outside the United States.
19	52	VoIP/Wireless Lines	Designates the quantity of numbers a VoIP customer is using. (Taxable amount and number of lines are irrelevant for this service type. Tax is calculated based on the number of transactions passed. For two lines you would pass two transactions.) This service type will return E911 at the wireless rate regardless of whether it is paired with the VoIP or VoIP/A transaction type.
19	53	VoIP/LNP	Fixed, monthly charge associated with transferring an existing phone number to a VoIP service provider.
19	566	VoIP/PBX Outbound Channel	Designates the number of voice grade communications channels leaving a subscriber's premises through a VoIP PBX connecting the subscriber's premises to the public switched network. Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of outbound channels designated in the lines field. Used in Conjunction with 19/41 and 19/578.
19	577	VoIP/Enhanced Features	Charges and fees for additional feature charges of VoIP services which are separate from voice transmission related features as defined by the FCC. (Includes services such as voicemail, interactive voice response, audiotext information services, and protocol processing.)
19	578	VoIP/PBX	Designates the number of PBX trunks a VoIP customer is using. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of trunks designated in the lines field. Used in conjunction with 19/41 and 19/566.)

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
19	579	VoIP/PBX High Capacity	Designates the number of High Capacity Trunks a customer is using. Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of High Capacity Trunks designated in the lines field. Used in conjunction with 19/580 and 19/582.
19	580	VoIP/High Capacity Extension	Designates the number of VoIP extensions a VoIP service customer is using on a High Capacity Trunk. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of extensions designated in the lines field.) Designates the number of VoIP extensions a VoIP service customer is using on a High Capacity Trunk. Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of extensions designated in the lines field. Used in conjunction with 19/579 and 19/582.
19	582	VoIP/High Capacity Outbound Channel	Designates the number of voice grade communications channels leaving a subscriber's premises through a VoIP High Capacity Trunk connecting the subscriber's premises to the public switched network. Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of outbound channels designated in the lines field. Used in conjunction with 19/579 and 19/580.
19	596	VoIP/Access-Local Only Service	Basic monthly flat rate charge for Local Only Service VoIP.
19	635	VoIP/Toll Free Number	Monthly recurring charge for access to a VoIP toll free number.
20	6	VoIP/Access Charge	Basic monthly flat rate for VoIP service.
20	8	VoIP/Install	Charge for installation of VoIP services.
20	11	VoIP/Activation	One-time charges for activating a VoIP account. (Mutually exclusive of the other VoIP charges.)
20	13	VoIP/Equipment Repair	Charge for repair of equipment necessary to make VoIP calls.
20	14	VoIP/Late Charge	Category for late charges that were originally taxed using one of the VoIP (20) transaction categories.
20	21	VoIP/Lines	Designates the quantity of numbers a VoIP customer is using. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of lines designated in the lines field.) This service type will return E911 at the landline rate regardless of whether it is paired with the VoIP or VoIP transaction type.
20	30	VoIP/Feature Charge	Charges and fees for additional feature charges of VoIP services. (Includes services such as call waiting, caller ID, call blocking, call forwarding, etc.)
20	37	VoIP/Equipment Rental	Charge for renting equipment necessary to make VoIP phone calls.

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
20	41	VoIP/A/PBX Extension	Designates the number of VoIP PBX extensions a VoIP service customer is using. Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of extensions designated in the lines field. Used in conjunction with 20/566 and 20/578.
20	43	VoIP/A/Invoice	Mapping category for transactions on a per invoice basis. (Tax is based per invoice per account per billing cycle. Taxable amount or numbers of lines are irrelevant for this transaction/service type.)
20	48	VoIP/A/Wireless Access Charge	This will tax similar to Cellular/Access Charge. Federal and State USF are applied, but at the wireless safe harbor rate.
20	49	VoIP/A/Interstate Usage	Portion of MRC, or per-minute charges, attributable to calls that cross state lines but do not leave the United States.
20	50	VoIP/A/Intrastate Usage	Portion of MRC, or per-minute charges, attributable to calls that do not cross state lines.
20	51	VoIP/A/International Usage	Portion of MRC, or per-minute charges, attributable to calls that originate inside the United States and terminate outside the United States.
20	52	VoIP/A/Wireless Lines	Designates the quantity of numbers a VoIP customer is using. (Taxable amount and number of lines are irrelevant for this service type. Tax is calculated based on the number of transactions passed. For two lines you would pass two transactions.) This service type will return E911 at the wireless rate regardless of whether it is paired with the VoIP or VoIP/A transaction type.
20	53	VoIP/A/LNP	Fixed, monthly charge associated with transferring an existing phone number to a VoIP service provider.
20	566	VoIP/A/PBX Outbound Channel	Designates the number of voice grade communications channels leaving a subscriber's premises through a VoIP PBX connecting the subscribers premises to the public switched network. Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of outbound channels designated in the lines field. Used in conjunction with 20/41 and 20/578.
20	577	VoIP/A/Enhanced Features	Charges and fees for additional feature charges of VoIP services which are separate from basic transmission services. (Includes services such as voicemail, interactive voice response, audiotext information services, and protocol processing.)
20	578	VoIP/A/PBX	Designates the number of PBX trunks a VoIP customer is using. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of trunks designated in the lines field. Used in conjunction with 20/41 and 20/566.)

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
20	579	VoIP/A/PBX High Capacity	Designates the number of High Capacity Trunks a customer is using. Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of High Capacity Trunks designated in the lines field. Used in conjunction with 20/580 and 20/582.
20	580	VoIP/A/High Capacity Extension	Designates the number of VoIP extensions a VoIP service customer is using on a High Capacity Trunk. Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of extensions designated in the lines field. Used in conjunction with 20/579 and 20/582.
20	582	VoIP/A/High Capacity Outbound Channel	Designates the number of voice grade communications channels leaving a subscriber's premises through a VoIP High Capacity Trunk connecting the subscriber's premises to the public switched network. Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of High Capacity Outbound Channels designated in the lines field. Used in conjunction with 20/579 and 20/580.
20	596	VoIP/A/Access-Local Only Service	Basic monthly flat rate charge for Local Only Service VoIP.
20	635	VoIP/A/Toll Free Number	Monthly recurring charge for access to a VoIP toll free number.
21	21	Payphone/Lines	Line charges for provisioning of service to a coin operated phone. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of lines designated in the lines field.)
21	49	Payphone/Interstate Usage	Charges for calls that cross state boundaries from a coin operated phone.
21	50	Payphone/Intrastate Usage	Charges for calls that do not cross state boundaries from a coin operated phone.
21	55	Payphone/Local Usage	Charges for local calls from a coin operated phone.
21	56	Payphone/Provisioning	MRC related charges for the provisioning of service to a coin-operated phone.
24	59	Software/Licensed Software	An agreement for the use of software for a specified period.
24	60	Software/Software Maintenance Agreement	A contract that covers the contract holder for the expense of maintaining and updating software.
24	61	Software/Report on CD or Paper Form	Report generated and provided to end user delivered on CD or paper.
24	595	Software/Downloaded Licensed Software	An agreement for the use of software for a specified period. Transferable to the computer by electronic means.
24	636	Software/Remotely Accessed Software	A service that provides access and usage of software that remains in the possession of the seller and is remotely accessed by a customer. If data is manipulated by the software, it is user created data.
25	62	Timesharing/Information Retrieval	Access to a computer through a remote terminal that allows retrieval of stored data created by the user.

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
25	646	Timesharing/Information Retrieval (Provider Data)	Access to a computer through a remote terminal that allows retrieval of stored data created by the service provider.
58	563	Dark Fiber Lease/Facilities	Lease of Dark Fiber installed on property owned by the lessor.
58	564	Dark Fiber Lease/Non-Facilities	Lease of Dark Fiber installed on property not owned by the lessor.
58	604	Dark Fiber Lease Facilities/Local Svc	Lease of Dark Fiber installed on property owned by the lessor used for local telecommunications service
58	605	Dark Fiber Lease/Non-Facilities-Local Svc	Lease of Dark Fiber installed on property not owned by the lessor used for local telecommunications service
59	6	VoIP Nomadic/Access Charge	Basic monthly flat rate for VoIP service.
59	8	VoIP Nomadic/Install	Charge for installation of VoIP services.
59	11	VoIP Nomadic/Activation	One-time charges for activating a VoIP account. (Mutually exclusive of the other VoIP charges.)
59	13	VoIP Nomadic/Equipment Repair	Charge for repair of equipment necessary to make VoIP calls.
59	14	VoIP Nomadic/Late Charge	Category for late charges that were originally taxed using one of the VoIP Nomadic transaction categories.
59	21	VoIP Nomadic/Lines	Designates the quantity of numbers a VoIP customer is using. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of lines designated in the lines field.)
59	30	VoIP Nomadic/Feature Charge	Charges and fees for additional feature charges of VoIP services. (Includes services such as call waiting, caller ID, call blocking, call forwarding, etc.)
59	37	VoIP Nomadic/Equipment Rental	Charge for renting equipment necessary to make VoIP phone calls.
59	41	VoIP Nomadic/PBX Extension	Designates the number of VoIP PBX extensions a local service customer is using. Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of extensions designated in the lines field. Used in conjunction with 59/566 and 59/578.
59	43	VoIP Nomadic/Invoice	Mapping category for transactions on a per invoice basis. (Tax is based per invoice per account per billing cycle. Taxable amount or numbers of lines are irrelevant for this transaction/service type.)
59	49	VoIP Nomadic/Interstate Usage	Portion of MRC, or per-minute charges, attributable to calls that cross state lines but do not leave the United States.
59	50	VoIP Nomadic/Intrastate Usage	Portion of MRC, or per-minute charges, attributable to calls that do not cross state lines.
59	51	VoIP Nomadic/International Usage	Portion of MRC, or per-minute charges, attributable to calls that originate inside the United States and terminate outside the United States.
59	53	VoIP Nomadic/LNP	Fixed, monthly charge associated with transferring an existing phone number to a VoIP service provider.

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
59	566	VoIP Nomadic/PBX Outbound Channel	Designates the number of voice grade communications channels leaving a subscriber's premises through a VoIP PBX connecting the subscriber's premises to the public switched network. Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of outbound channels designated in the lines field. Used in conjunction with 59/41 and 59/578.
59	577	VoIP Nomadic/Enhanced Features	Charges and fees for additional feature charges of VoIP services which are separate from basic transmission services. (Includes services such as voicemail, interactive voice response, audiotext information services, and protocol processing.)
59	578	VoIP Nomadic/PBX	Designates the number of PBX trunks a VoIP customer is using. (Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of trunks designated in the lines field. Used in conjunction with 59/41 and 59/566.)
59	579	VoIP Nomadic/PBX High Capacity	Designates the number of High Capacity Trunks a customer is using. Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of High Capacity Trunks designated in the lines field. Used in conjunction with 59/580 and 59/582.
59	580	VoIP Nomadic/High Capacity Extension	Designates the number of VoIP extensions a VoIP service customer is using on a High Capacity Trunk. Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of extensions designated in the lines field. Used in conjunction with 59/579 and 59/582.
59	582	VoIP Nomadic/High Capacity Outbound Channel	Designates the number of voice grade communications channels leaving a subscriber's premises through a VoIP High Capacity Trunk connecting the subscriber's premises to the public switched network. Taxable amount is irrelevant for this transaction/service type. Tax is calculated based on the number of outbound channels designated in the lines field. Used in conjunction with 59/579 and 59/580.
59	635	VoIP Nomadic/Toll Free Number	Monthly recurring charge for access to a VoIP Nomadic toll free number.
60	10	Satellite Phone/Usage	Satellite per minute and/or per use charges. Avalara has added this combination for development and testing purposes. Avalara plans that the combination will be available for use in tax production approximately the first of the year.
61	585	VPN/Interstate MPLS	Charge for Interstate Virtual Private Network using MPLS.
61	586	VPN/Intrastate MPLS	Charge for Intrastate Virtual Private Network using MPLS.
61	650	VPN/MPLS Intrastate Activation	One-time charge for the activation of an intrastate virtual private network (VPN) using multiprotocol label switching (MPLS). It is mutually exclusive of the other VPN charges.

Table 2-1 Valid Transaction / Service Pairs

IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
61	651	VPN/MPLS Install	One-time charge for the installation, administration, modification, or termination of a virtual private network (VPN) using multiprotocol label switching (MPLS). This should be used for install fee, change order fees, add-service fees, and termination account/service fees, but not for repair/service fees.
61	652	VPN/MPLS Service	One-time charge for manual, physical service to a virtual private network (VPN) using multiprotocol label switching (MPLS), such as a truck roll to the customer premise. All manual repair services should fall into this category. This designation should not be used for administrative fees or service change fees.
61	653	VPN/MPLS Interstate Activation	One-time charge for the activation of an interstate virtual private network (VPN) using multiprotocol label switching (MPLS). It is mutually exclusive of the other VPN charges.
64	648	Conferencing/Intrastate with FCC Jurisdiction	Per-minute and per-participant charges for dial-in service provided in conjunction with connecting conference participants utilizing a conferencing bridge when all participants are located within one state. Intended to be used to return Federal charges with the State charges.
64	649	Conferencing/Interstate without FCC Jurisdiction	Per-minute and per-participant charges for dial in service provided in conjunction with connecting conference participants utilizing a conferencing bridge when all participants are located in different states. Intended to be used to return State charges for conferencing without Federal charges.
65	6	Non-Interconnected VoIP/Access Charge	Basic monthly flat rate for non-interconnected VoIP service.
65	11	Non-Interconnected VoIP/Activation	One-time charges for activating a non-interconnected VoIP account. (Mutually exclusive of the other VoIP charges.)
65	14	Non-Interconnected VoIP/Late Charge	Category for late charges that were originally taxed using one of the "Non-Interconnected VoIP" transaction categories.
65	30	Non-Interconnected VoIP/Feature Charge	Charges and fees for additional feature charges of non-interconnected VoIP services. (Includes services such as call waiting, caller ID, call blocking, call forwarding, etc.)
65	43	Non-Interconnected VoIP/Invoice	Mapping category for non-interconnected VoIP transactions on a per invoice basis. (Tax is based per invoice per account per billing cycle. Taxable amount or numbers of lines are irrelevant for this transaction/service type.)
65	49	Non-Interconnected VoIP/Interstate Usage	Portion of MRC, or per-minute charges, attributable to non-interconnected VoIP calls that cross state lines but do not leave the United States.
65	50	Non-Interconnected VoIP/Intrastate Usage	Portion of MRC, or per-minute charges, attributable to non-interconnected VoIP calls that do not cross state lines.

Table 2-1 Valid Transaction / Service Pairs			
IDENTIFIER		NAME	DESCRIPTION
Trans Type	Svc Type		
65	51	Non-Interconnected VoIP/International Usage	Portion of MRC, or per-minute charges, attributable to non-interconnected VoIP calls that originate inside the United States and terminate outside the United States.
65	577	Non-Interconnected VoIP/Enhanced Features	Charges and fees for additional feature charges of non-interconnected VoIP services which are separate from basic transmission services. (Includes services such as voicemail, interactive voice response, audio text information services, and protocol processing.)

2.2.1 File transervdescsau.txt

A file named transervdescsau.txt is present in the database directory in each month's update. This file contains all of the valid transaction type/service type pairs, with the description of those types and of the pairing. It is a fixed format file, though there are spaces and commas between some of the fields for readability. It is described in the table below.

transervdesc.txt File Format Key	
Description	Columns
Alternate Flag	1-1
Market Number	3-4
Market Name	6-35
Transaction Type Number	38-39
Transaction Type Description	41-90
Service Type Number	93-95
Service Type Description	97-146
Long Description	148-end of line

2.3 Transaction Mapping Guidelines

2.3.1 Long Distance Transaction and Service Types

The long distance transaction and service types provide the ability to tax 1+, 0+, 800, 900, WATS and private lines for both interstate and intrastate calls. The capability to tax access charges, late charges, services, installations and International USA Inbound transactions is also provided.

Refer to Table 2-3 for a list of combined transaction and service types that are used for taxing long distance transactions.

Table 2-3 Long Distance Transaction and Service Types	
Transaction Types	Service Types
1 – Interstate	1 – Toll 2 – Toll-Free 3 – WATS 4 – Private Line 12 – International Toll

Table 2-3 Long Distance Transaction and Service Types

Transaction Types	Service Types
	14 – Late Charge 16 – 900 27 – Data 54 – Directory Assistance 562 – Local Loop 635 – Toll Free Number
2 – Intrastate	1 – Toll 2 – Toll-Free 3 – WATS 4 – Private Line 5 – Local Exchange 14 – Late Charge 16 – 900 27 – Data 54 – Directory Assistance 630 – Private Line (10% Rule) 631 – Data (10% Rule) 635 – Toll Free Number
3 – Other	6 – Access Charge 6 – Local Loop 9 – Directory Ads 14 – Late Charge 34 – Conference Bridge 37 – Equipment Rental 38 – Wire Maintenance 46 – PICC 47 – No Pick PICC 57 – Data Processing 96 – No Pick PICC Bundle 97 – PICC Bundle 570 – Directory Listing 575 – Conference Bridge- Intrastate 576 – Conference Bridge-Intrastate w Dial-in 589 – Conference Bridge-Interstate 593 – Info Svcs-Private Physical Trans 594 – Info Svcs-Private Electronic Trans 597 – Info Svcs-Public-Electronic Trans 598 – Info Svcs-Public Physical Trans 599 – E-mail hosting service 600 – Real Property Rental 602 – Services-Professional 603 – Online Services 608 – Conference Bridge Interstate w Dial-In 614 – Telecom Equipment Rental 632 – Service Contracts 638 – Security Monitoring Services

Table 2-3 Long Distance Transaction and Service Types	
Transaction Types	Service Types
	639 – Streaming Internet Video 644 – Info Svcs-Pub Elec Trans (Fin & Securities)
4 – Non-recurring	7 – Service 8 – Install 8 – Local Install 11 – Local Activation 14 – Late Charge
14 – International	25 – USA Inbound

2.3.2 Wireless Transaction and Service Types

Refer to Table 2-4 for a list of combined transaction and service types that are provided for taxing cellular and PCS long distance transactions.

Table 2-4 Wireless Transaction and Service Types	
Transaction Types	Service Types
13 – Cellular	6 – Access Charge 10 – Usage 11 – Activation 14 – Late Charge 30 – Feature Charge 33 – Roaming Charge 49 – Interstate Usage 50 – Intrastate Usage 98 – Access number 99 – Interstate Access Charge 100 – Intrastate Access Charge 101 – Interstate Roaming 102 – Intrastate Roaming 572 – Digital Download 577 – Enhanced Features 591 – Access Charge-No Contract 592 – Access Number-No Contract 610 – Early Termination Fees 622 – Text Message

2.3.3 Enhanced Service Transaction and Service Types

These transaction and service types allow the taxing of internet, paging, FAX, and voice mail services. Refer to Table 2-5 for a list of combined transaction and service types that are typically used for taxing enhanced service transactions.

Table 2-5 Enhanced Service Transaction and Service Types	
Transaction Types	Service Types
6 – Paging	6 – Access Charge 10 – Usage

	11 – Activation 13 – Equipment Repair
8 – Fax	10 – Usage
9 – Voice mail	6 – Access Charge 10 – Usage 11 – Activation 14 – Late Charge

2.3.4 Internet Transaction and Service Types

The federal government and most states do not tax Internet usage. AFC will determine this based upon the jurisdiction information provided. If taxable, AFC applies the appropriate taxes for the given jurisdiction.

Refer to Table 2-6 for a list of combined transaction and service types that are used for taxing Internet usage.

Table 2-6 Internet Transaction and Service Types	
Transaction Types	Service Types
5 – Internet	6 – Access Charge 29 – Web Hosting 58 – Access Line

2.3.5 Local Service Transaction and Service Types

Local Service mapping demands special attention because more than one Transaction/Service Type is required to generate all of the necessary rated and per line taxes. Refer to Table 2-7 for a list of combined transaction and service types that are associated with local service.

Table 2-7 Local Service Transaction and Service Types	
Transaction Types	Service Types
7 – Local	4 – Private Line 5 – Local Exchange 14 – Late Charge 20 – FCC Subscriber Line Fee 20 – Number Portability Recovery 21 – Lines 24 – PBX/Trunk 27 – Data 30 – Local Feature Charge 40 – Centrex / DID Extension 41 – PBX Extension 42 – Local Trunk(Alias as Local Centrex) 43 – Invoice 45 – High Capacity Trunk 84 – Late Charge Bundle 85 – Local Exchange Bundle 86 – FCC Subscriber Line Fee Bundle 86 – Number Portability Recovery Bundle

Table 2-7 Local Service Transaction and Service Types

Transaction Types	Service Types
	87 – Lines Bundle 89 – PBX Trunk Bundle 90 – Local Feature Charge Bundle 91 – Centrex Extension Bundle 92 – PBX Extension Bundle 93 – Centrex Trunk Bundle 94 – Invoice Bundle 95 – High Capacity Trunk Bundle 566 – PBX Outbound Channel 567 – PBX Outbound Channel Bundle 580 – High Capacity Extension 581 – High Capacity Extension Bundle 582 – High Capacity Outbound Channel 583 – High Capacity Outbound Channel Bundle 587 – Centrex Outbound Channel 588 – Centrex Outbound Channel Bundle 612 – FCC Subscriber Line Fee Multi Line 613 – FCC Subscriber Line Fee Multi Line Bundle 623 – Centrex Invoice 625 – Customer Premise Equip Rental 641 – FCC Subscriber Line Charge Centrex 642 – FCC Subscriber Line Charge Centrex Bundle

The following transaction/service combinations should be passed to AFC in order to capture all local taxes.

2.3.5.1 Required Transactions to Return Rated Charges

Taxation of Local Exchange starts with passing five required transactions.

1. Local / Local Exchange (7/5) – Charges for the monthly recurring charge, usage, local loop, local flat rates, and other similar charges for local telecom service. (Due to unique Federal Taxation the FCC Subscriber Line Fee and Local Number Portability must be distinguished.)
2. Local / FCC Subscriber Fee (7/20) – Charge for assessing the carrier's subscriber line charge to a local service customer.
3. Local / LNP (7/20) – Fixed monthly charge through which local telephone companies may recover some of the costs associated with providing local number portability service. (Both the 7/05 and 7/20 are sent in with the corresponding charge. The lines fields must be set to zero.)
4. Local / Invoice (7/43) – This combination returns taxes assessed on a per-invoice basis. The charge and lines fields are irrelevant and must be set to zero. Pass one transaction for each invoice.

5. Local / Data (7/27) - Local data service charge, MRC, and other related service type charges and features. (This combination is appropriate for transmissions that carry data exclusively. Use Private Line if any component is voice. This is a recommended mapping for data transmissions carried over DSL, ATM, T-1, frame relay lines and other non-voice services. This is not for Internet Access provided over DSL lines.)

2.3.5.2 Required Transactions to Return Per-Line Charges

Not all of the necessary telecom taxes are returned using transaction/service combinations that return rated charges. There exist per-line taxes, such as E911, that are returned using line transactions. It is necessary to select one of the basic categories depending on the customer. These line transactions fall into one of the following the three categories.

1. Local / Lines (7/21) – Designates the number of lines a local service customer is using. This can be for either a residential or business customer who has a simple wire line connection into the premises.
2. Local / High Capacity Trunk (7/45) – Designates the number of High Capacity Trunks a customer is using. High Capacity Trunks are usually defined as T1 or greater. (The taxable amount for the 7/21 and 7/45 transactions are irrelevant and must be populated with a zero dollar amount. Only the lines data should be populated with the total number of lines or high capacity trunks to be taxed.)
3. Business Customers with PBX or Centrex

In some cases, a business customer may have a PBX or Centrex instead of one or two lines. This situation requires two transactions to be passed into the system in order to return the proper per line taxes.

Some jurisdictions assess per-line taxes, such as E911, on the number of PBX or Centrex trunks used. Others assess per-line taxes on the number of PBX or Centrex DID Extensions. These taxes are usually assessed at a reduced rate. Still others assess taxes on both. (For example, they may assess the TRS on the trunk, but want an E911 for each extension.)

- a. Business Customers with PBX require these two transactions to handle taxation:

Local / PBX Trunk (7/24) – Designates the number of PBX trunks a local service customer is using. The lines field of the 7/24 transactions should be populated with the number of PBX trunks to be taxed.

Local / PBX Extensions (7/41) – Designates the number of PBX extensions a local service customer is using. The lines field of the 7/41 transactions should be populated with the number of PBX extensions to be taxed. (The taxable amount for

the 7/24 and 7/41 transactions are irrelevant and must be populated with a zero dollar amount.)

- b. Business Customers with Centrex require these two transactions to handle taxation:

Local / Centrex Trunk (7/42) – Designates the number of Centrex trunks a local service customer is using. The lines field of the 07/42 transactions should be populated with the number of Centrex trunks to be taxed.

Local / Centrex DID Extensions (7/40) – Designates the number of Centrex extensions a local service customer is using. The lines field of the 7/40 transactions should be populated with the number of Centrex extensions to be taxed. (The taxable amount for the 7/42 and 7/40 transactions are irrelevant and must be populated with a zero dollar amount.)

2.3.6 Sales Transaction and Service Types

The transaction type “Sales” and the service type “Product” is provided for the taxing the sale of finished goods. All of the sales tax rates for the various states, counties and localities in the United States are provided.

The transaction type “Sales” and the service type “Use” is provided for taxing the use of products. All of the sales tax rates for the various states, counties and localities in the United States are provided.

NOTE:

At this time, AFC does not handle sales tax exceptions. If a transaction is supplied to AFC using this transaction and service type, sales taxes will be applied based upon the jurisdiction. If sales taxes do not apply to the specified transaction, it is the user's responsibility to determine this and not pass the transaction to AFC.

Refer to Table 2-8 or a list of combined transaction and service types that are associated with Sales.

Table 2-8 Sales Transaction and Service Types	
Transaction Types	Service Types
10 – Sales	15 – Product 31 – Use 32 – Debit 63 – Restocking Fee – Rental 64 – Restocking Fee – Purchase 65 – Partial Credit 103 – Sales Tax and FUSF 565 – Debit-Wireless 568 – Central Office Equipment – Sales 569 – Central Office Equipment – Use 643 – Debit-Wireless (Indirect Non-Carrier Sale)

2.3.6.1 Debit Transactions

Debit transactions involve two transaction types; one at the time of sale as prepayment for the service, the other at the times of use following the purchase.

The time of sale payment should be processed using the Sales/Debit (10/32) transaction. The transactions that follow the sale as the time on the card is expended will be either the Interstate Call (1/1) transaction or the Intrastate Call (2/1) transaction.

When using API's, the initial Sales/Debit transaction of the prepaid calling card would be processed through the normal API calls such as EZTaxPCodeEx. The Toll call transactions would be processed through the Debit API calls.

2.3.7 Shipping Transaction and Service Types

Shipping Taxes are provided for the United States, its counties and localities. Refer to for a list Table 2-9 of combined transaction and service types that are associated with shipping taxes.

Table 2-9 Shipping Transaction and Service Types	
Transaction Types	Service Types
11 – Shipping	17 – FOB Origin 18 – FOB Destination

2.3.8 Natural Gas Transaction and Service Types

Refer to Table 2-10 for a list of combined transaction and service types that are associated with Natural Gas taxes.

Table 2-10 Natural Gas Transaction and Service Types	
Transaction Types	Service Types
12 – Natural gas	6 – Access Charge 19 – Consumption

2.3.9 Telephony Transaction and Service Types

Refer to the table below for combined transaction and service types that are associated with Telephony Services.

Telephony Transaction and Service Types	
Transaction Types	Service Types
15 – Telephony	7 – Service 624 – Wireless Service

	627 – Internet Access 629 – Messaging Services
--	---

Refer to the table below for a list of countries which should use the Telephony/Service T/S pair.

Countries Using Telephony/Service	
Country	ISO Code
ANTIGUA	ATG
ARGENTINA	ARG
AUSTRALIA	AUS
AUSTRIA	AUT
AZERBAIJAN	AZE
BAHAMAS	BHS
BAHRAIN	BHR
BARBADOS	BRB
BELGIUM	BEL
BELIZE	BLZ
BERMUDA	BMU
BOLIVIA	BOL
BOSNIA AND HERZEGOVINA	BIH
BRAZIL	BRA
BULGARIA	BGR
CAMBODIA	KHM
CANADA	CAN
CHILE	CHL
CHINA	CHN
COLOMBIA	COL
COSTA RICA	CRI
CROATIA	HRV
CYPRUS	CYP
CZECH REPUBLIC	CZE
DENMARK	DNK
DOMINICA	DMA
DOMINICAN REPUBLIC	DOM
ECUADOR	ECU
EGYPT	EGY
EL SALVADOR	SLV
ESTONIA	EST
FIJI	FJI
FINLAND	FIN
FRANCE	FRA
GERMANY	DEU
GHANA	GHA
GREECE	GRC
GRENADA	GRD
GUATEMALA	GTM
GUYANA	GUY

Countries Using Telephony/Service	
Country	ISO Code
HONDURAS	HND
HUNGARY	HUN
ICELAND	ISL
INDIA	IND
INDONESIA	IDN
IRELAND	IRL
ISRAEL	ISR
ITALY	ITA
JAMAICA	JAM
JAPAN	JPN
KENYA	KEN
KUWAIT	KWT
LATVIA	LVA
LITHUANIA	LTU
LUXEMBOURG	LUX
MALAYSIA	MYS
MAURITIUS	MUS
MEXICO	MEX
MOROCCO	MAR
NAMIBIA	NAM
NETHERLANDS	NLD
NEW ZEALAND	NZL
NICARAGUA	NIC
NIGERIA	NGA
NORWAY	NOR
OMAN	OMN
PANAMA	PAN
PAPUA NEW GUINEA	PNG
PERU	PER
PHILIPPINES	PHL
POLAND	POL
PORTUGAL	PRT
QATAR	QAT
REPUBLIC OF MALTA	MLT
REUNION ISLAND	REU
ROMANIA	ROM
RUSSIA	RUS
SAUDI ARABIA	SAU
SERBIA	SRB
SENEGAL	SEN
SINGAPORE	SGP
SLOVAK REPUBLIC	SVK
SLOVENIA	SVN
SOUTH AFRICA	ZAF
SOUTH KOREA	KOR
SPAIN	ESP

Countries Using Telephony/Service	
Country	ISO Code
ST KITTS AND NEVIS	KNA
ST VINCENT	VCT
SWEDEN	SWE
SWITZERLAND	CHE
TAIWAN	TWN
TANZANIA	TZA
THAILAND	THA
TRINIDAD AND TOBAGO	TTO
TURKEY	TUR
UGANDA	UGA
UKRAINE	UKR
UNITED ARAB EMIRATES	ARE
UNITED KINGDOM	GBR
UNITED STATES OF AMERICA	USA
URUGUAY	URY
VENEZUELA	VEN
VIETNAM	VNM
ZAMBIA	ZMB

2.3.10 Cable Television Transaction and Service Types

Refer to Table 2-11 for a list of combined transaction and service types that are associated with Cable Television.

Table 2-11 Cable Television Transaction and Service Types	
Transaction Types	Service Types
16 – Cable Television	6 – Access Charge 8 – Install 13 – Equipment Repair 14 – Late Charge 35 – Premium Service 36 – Pay Per View Service 37 – Equipment Rental 39 – TV Guide 584 – Digital Channel Tier 610 – Early Termination Fees 615 - Equipment Sales

2.3.11 Satellite Television Transaction and Service Types

Refer to Table 2-12 for a list of combined transaction and service types that are associated with Satellite Television.

Table 2-12 Satellite Television Transaction and Service Types

Transaction Types	Service Types
18 – Satellite Television	6 – Access Charge 8 – Install 13 – Equipment Repair 14 – Late Charge 35 – Premium Service 36 – Pay Per View Service 37 – Equipment Rental 39 – TV Guide

2.3.12 Voice over Internet Protocol (VoIP) Transaction and Service Types

Nomadic VoIP service (such as soft phone service) should be handled using the Nomadic VoIP transaction type. Fixed location VoIP service (such as a business PBX system) faces different taxation decisions from State, County and Local jurisdictions.

Some jurisdictions have indicated that their taxes apply to VoIP, some have stated that they are not taxing VoIP and don't plan to in the near future and others are taking a wait and see stance.

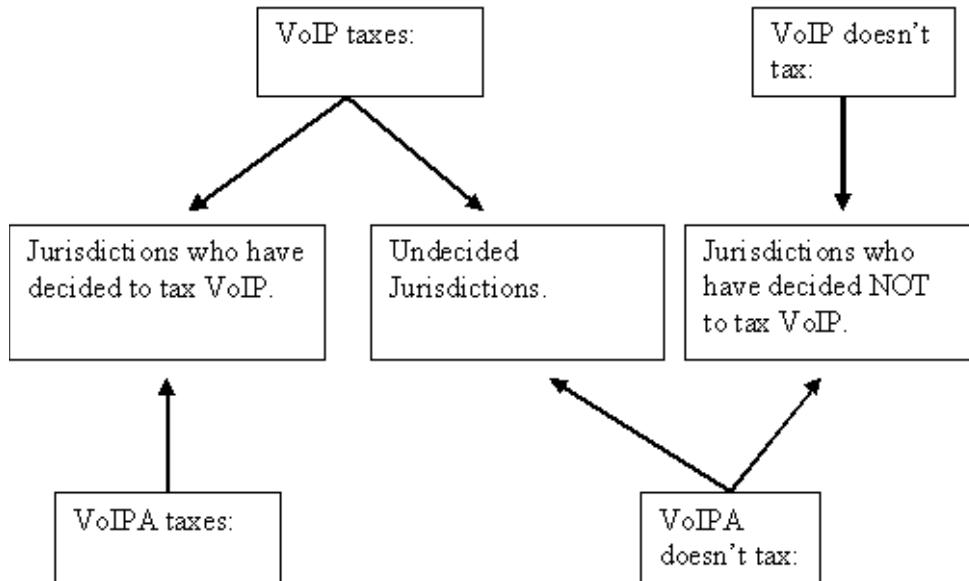
Because of these jurisdiction inconsistencies, providers are forced to take different attitudes toward taxation. Some are taxing VoIP as they would normal POTS lines while others are taking the position that VoIP is not taxable as a telecom service. Avalara's versatile taxation engine is designed to support both of these approaches with the transaction types VoIP and VoIPA.

Refer to Table 2-6 and Figure 2-1. The VoIP transaction type returns telecom taxes in a jurisdiction that has stated that their taxes apply to VoIP and jurisdictions that are undecided. The VoIPA transaction type will only return telecom taxes in jurisdictions that have definitively stated their taxes apply to VoIP. These two approaches will converge as jurisdictions define how they want VoIP taxed.

Table 2-13 VoIP Transaction Returns

	VoIP	VoIPA
Jurisdictions who tax VoIP	Tax	Tax
Undecided Jurisdictions	Tax	Don't Tax
Jurisdictions who don't tax VoIP	Don't Tax	Don't Tax

Figure 2-1 VoIP Tax



2.3.12.1 E911 and other Per-Line Taxes

Since VoIP users can connect via a landline or wireless connection and some jurisdictions have different rates for each of these, both rates are incorporated into the AFC system.

Use the “Lines” service type to receive the landline rate and the “Wireless Lines” service type to receive the wireless rate. Application of any per-line taxes, such as State TRS or State USF, is dependent on whether the VoIP or VoIPAA is used.

The resulting four choices are:

- a. VoIP / Lines (19/21) – Returns landline E911 and per-line telecom taxes in jurisdictions that tax VoIP, and jurisdictions that are taking a wait and see approach. This combination will return only E911 at the landline rate in jurisdictions that have stated they are not taxing VoIP.
- b. VoIP / Wireless Lines (19/52) – Will return wireless E911 and per-line telecom taxes in jurisdictions that tax VoIP, and jurisdictions that are taking a wait and see approach. This combination will only return E911 at the wireless rate in jurisdictions that have stated they are not taxing VoIP.
- c. VoIPAA / Lines (20/21) – Will return only landline E911 in jurisdictions that have stated they don’t tax VoIP, and in jurisdictions that are undecided. Will return telecom taxes and landline E911 in jurisdictions that tax VoIP.

- d. VoIP / Wireless Lines (20/52) – Will return only wireless E911 in jurisdictions that have stated they don't tax VoIP, and in jurisdictions that are undecided. Will return telecom taxes and wireless E911 in jurisdictions that tax VoIP.

NOTE:

The per-line taxes are returned as a “fixed” amount when using the Wireless Lines service type. When passing the Wireless Lines service type you do not need to enter the taxable amount or number of lines. You only need to pass one transaction for each line that needs to be taxed.

2.3.12.2 VoIP Transaction and Service Types

Refer to Table 2-14 for a list of combined transaction and service types that are used with VoIP transaction types.

Table 2-14 VoIP Transaction and Service Types	
Transaction Types	Service Types
19 – VoIP	6 – Access Charge 8 – Install 11 – Activation 13 – Equipment Repair 14 – Late Charge 21 – Lines 30 – Feature Charge 37 – Equipment Rental 41 – PBX Extension 43 – Invoice 48 – Wireless Access Charge 49 – Interstate Usage 50 – Intrastate Usage 51 – International Usage 52 – Wireless Lines 53 – LNP 566 – PBX Outbound Channel 577 – Enhanced Features 578 – PBX 579 – PBX High Capacity 580 – High Capacity Extension 582 – High Capacity Outbound 596 – Access-Local Only Service 635 – Toll Free Number

2.3.12.3 VoIP Transaction and Service Types

Refer to Table 2-15 for a list of combined transaction and service types that are used with VoIP transaction types.

Table 2-15 VoIP Transaction and Service Types

Transaction Types	Service Types
20 – VoIPA	6 – Access Charge 8 – Install 11 – Activation 13 – Equipment Repair 14 – Late Charge 21 – Lines 30 – Feature Charge 37 – Equipment Rental 41 – PBX Extension 43 – Invoice 48 – Wireless Access Charge 49 – Interstate Usage 50 – Intrastate Usage 51 – International Usage 52 – Wireless Lines 53 – LNP 566 – PBX Outbound Channel 577 – Enhanced Features 578 – PBX 579 – PBX High Capacity 580 – High Capacity Extension 582 – High Capacity Outbound Channel 596 – Access-Local Only Service 635 – Toll Free Number

2.3.12.4 Nomadic VoIP Transaction and Service Types

Refer to Table 2-16 for a list of combined transaction and service types that are used with Nomadic VoIP transaction types.

Table 2-16 Nomadic VoIP Transaction and Service Types	
Transaction Types	Service Types
59 – VoIP Nomadic	6 – Access Charge 8 – Install 11 – Activation 13 – Equipment Repair 14 – Late Charge 21 – Lines 30 – Feature Charge 37 – Equipment Rental 41 – PBX Extension 43 – Invoice 49 – Interstate Usage 50 – Intrastate Usage 51 – International Usage 53 – LNP 566 – PBX Outbound

Table 2-16 Nomadic VoIP Transaction and Service Types	
Transaction Types	Service Types
	577 – Enhanced Features 578 – PBX 579 – PBX High Capacity 580 – High Capacity 582 – High Capacity Outbound 635 – Toll Free Number

2.3.13 Payphone Transaction and Service Types

Refer to Table 2-17 for a list of combined transaction and service types that are used with Payphone transaction types.

Table 2-17 Payphone Transaction and Service Types	
Transaction Types	Service Types
21 – Payphone	21 – Lines 49 – Interstate Usage 50 – Intrastate Usage 55 – Local Usage 56 – Provisioning

2.3.14 Software Transaction and Service Types

Refer to Table 2-18 for a list of combined transaction and service types that are used with Software transaction types.

Table 2-18 Software Transaction and Service Types	
Transaction Types	Service Types
24 – Software	59 – Licensed Software 60 – Software Maintenance 61 – Report on CD or Paper Form 595 – Downloaded licensed software

2.3.15 Timesharing Transaction and Service Types

Refer to Table 2-19 for a list of combined transaction and service types that are used with Timesharing transaction types.

Table 2-19 Timesharing Transaction and Service Types	
Transaction Types	Service Types
25 – Timesharing	62 – Information Retrieval

2.3.16 Satellite Phone Transaction and Service Types

Refer to Table 2-20 for a list of combined transaction and service types that are used with Satellite Phone transaction types.

Table 2-20 Satellite Phone Transaction and Service Types	
Transaction Types	Service Types
60 – Satellite Phone	10 – Usage

2.3.17 VPN Transaction and Service Types

Refer to Table 2-21 for a list of combined transaction and service types that are used with VPN transaction types.

Table 2-21 VPN Transaction and Service Types	
Transaction Types	Service Types
61 – VPN	585 – Interstate MPLS 586 – Intrastate MPLS 650 – MPLS Instrastate Activation 651 – MPLS Install 652 – MPLS Service 653 – MPLS Interstate Activation

2.4 Tax Categories

Please reference the table below for a current list of Tax Categories.

Tax Categories	
Category ID	Name
0	No Category Description
1	Sales and Use Taxes
2	Business Taxes
3	Gross Receipts Taxes
4	Excise Taxes
5	Connectivity Charges
6	Regulatory Charges
7	E-911 Charges
8	Utility User Taxes
9	Right of Way Fees
10	Communications Services Tax
11	Cable Regulatory Fees

Tax Categories	
Category ID	Name
12	Reserved
13	Value Added Taxes

3. Transactions

AFC accepts transaction records from the user and returns calculated taxes. For AFC to return the correct taxation the information in the transaction records must be correct and in the format that AFC requires.

Each Transaction contains the following information:

1. Customer Transaction Information
 - a. Customer Information
 - b. Company Information
 - c. Transaction Data
2. Taxing Jurisdiction Information

The Customer Transaction Information has been divided into three sections as suggested above for the purpose of this discussion (see the table below).

Table 3-1 Customer Transaction Information

Customer Information	Company Information	Transaction Data
Business Class	Company Identifier	Charge
County Exempt	Facilities Based	Lines
County Exemption JCode	Franchise	Minutes
Customer Number (Primary Output Key)	Regulated/Unregulated	Sale/Resale
Customer Type		Service Type
Federal Exempt		Transaction Bill Date
Federal Exemption JCode		Transaction Type
Incorp		
Invoice Number		
Lifeline		
Local Exempt		
Locality Exemption JCode		
Service Class		
Service Level		
Tax Type Exemptions		
State Exempt		
State Exemption JCode		

Table 3-1 Customer Transaction Information

Customer Information	Company Information	Transaction Data
Tax Exempt		
Options		

3.1 Customer Information

Refer to the table below. The following customer information is contained in fields found in the transaction record.

Table 3-2 Customer Information

Description	Valid Entry
Business Class	CLEC = 1, ILEC = 0
Customer Number (Primary Output Key)	Customer number, user defined
Customer Type	0=residential, 1=business customer 2=Senior Citizen, 3=Industrial
Federal Exempt	If TRUE, transaction exempt from Federal Tax
State Exempt	If TRUE, transaction exempt from State Tax
County Exempt	If TRUE, transaction exempt from County Tax
Locality Exempt	If TRUE, transaction exempt from local tax
Incorp	TRUE indicates within incorporated area
Invoice Number	Invoice number (user defined)
Federal Exemption JCode	Jurisdiction for Federal exemption
State Exemption JCode	Jurisdiction for state exemption
County Exemption JCode	Jurisdiction for County exemption
Locality Exemption JCode	Jurisdiction for local exemption
Lifeline	Lifeline = TRUE, Non-lifeline = FALSE
Service Class	Primary Long Distance = 1, Local Service = 0, (Default Long Distance)
Service Level Number	Service level number (user defined)
Tax Type Exemptions	0 indicates no of special exemptions, other value indicates number of special exemptions
Tax Exempt	Pointer to tax exempt structure that contains the number of specific tax exemptions specified in tax type exemptions
Exempt Type *s_exempt	Reason for exemption
Optional Fields:	
Optional	User defined value for reporting
Optional Alpha	Optional alpha field
Optional 4	Optional Numeric field
Optional 5	Optional Numeric field
Optional 6	Optional Numeric field
Optional 7	Optional Numeric field
Optional 8	Optional Numeric field
Optional 9	Optional Numeric field
Optional 10	Optional Numeric field

[3.1.1 Business Class Indicator](#)

The Business Class Indicator field is used to specify if the business making the transaction is an Incumbent Local Exchange Company (ILEC) or a Competitive Local Exchange Company (CLEC).

- An ILEC company is engaged in selling services over company owned lines and equipment,
- A CLEC company is engaged in selling services competing with an incumbent provider.

[3.1.2 Customer Number \(Primary Output Key\)](#)

The Customer Number is a 20-character null terminated string field stored in the tax log and used as the Primary Output Key for all of the AFC billing reports.

[3.1.3 Customer Type](#)

This field is used to specify the type of customer involved in the transaction. The customer type is selected from one of the following four Customer Types.

- Business – When transactions are made at a place of business.
- Residential – When transactions are made by a customer for home use.
- Industrial – When transactions are made at an industrial business.
- Senior Citizen – When transactions are made by a customer meeting the jurisdiction requirements to be considered a senior citizen and qualify for senior citizen tax breaks.

[3.1.4 Exemption Levels](#)

The exemption level is the jurisdictional level of the taxing authority that defines the tax. It is used to exempt taxes at specific federal, state, county and/or local taxes.

[3.1.4.1 Federal Exempt](#)

The Federal Exempt field is used to specify a Federal level tax exemption.

Note: Most Federal taxes are only exempted when selling to a reseller who is registered, reporting, and remitting to the regulating agency. For this reason, a wholesale exemption or a tax type exemption must be used to exempt taxes at the Federal level.

[3.1.4.2 State Exempt](#)

The State Exempt field is used to specify a State level tax exemption.

[3.1.4.3 County Exempt](#)

The County Exempt field is used to specify a County level tax exemption.

3.1.4.4 Local Exempt

The Local Exempt field is used to specify a Local level tax exemption.

3.2 Incorp

The Incorp field is used to specify whether the customer involved in this transaction is inside or outside of the Local level designated as their location. The tax may or may not be affected by this designator depending upon whether or not the local level has taxes which would apply to the transaction/service type pair.

3.2.1 Invoice Number

The Invoice Number is an optional field used to aid users in uniquely identifying a billing record for a specific customer within their system.

3.2.2 JCode Exemption Levels

The exemption JCode is the JCode associated with the jurisdictional level of the taxing authority that defines the tax. It is used to exempt all federal, state, county and / or local taxes. If the exemption JCode fields are not specified then all taxes are exempt at that level.

To pass an individual tax exemption using the JCode Exemption, use the Tax Exempt structure to specify the tax type, tax level and the JCode for the jurisdiction. Refer to Section 4.1.1.2.

NOTE:

JCodes are an internal intermediate Jurisdiction Code that can change monthly. The JCode can be obtained from a PCode or an address using like functions.

3.2.2.1 Federal Exemption JCode

The Federal Exemption JCode field identifies the JCode of the jurisdiction (i.e. country) for a federal exemption.

3.2.2.2 State Exemption JCode

The State Exemption JCode field identifies the JCode of the jurisdiction for a state exemption.

3.2.2.3 County Exemption JCode

The County Exemption JCode field identifies the JCode of the jurisdiction for a county exemption.

3.2.2.4 Locality Exemption JCode

The Locality Exemption JCode field identifies the JCode of the jurisdiction for a locality exemption.

3.2.3 Lifeline Flag

The Lifeline Flag is used to indicate if a customer is a Lifeline participant.

3.2.4 Service Class Indicator

The Service Class Indicator is provided to delineate the Primary activity of an organization as either Long Distance or Local Service.

- Primary Long Distance providers are carriers vending their services with over 50% of the gross business activities in Long Distance revenue.
- Primary Local Service providers are carriers vending their services with over 50% of the gross business activities in Local Service revenue.

NOTE:

This has no effect on non-Telecom Transactions

3.2.5 Service Level Number

The Service Level Number is an optional field, commonly populated with the transaction type and service type. This information is stored in the tax log. Some AFC reporting utilities use this field when sorting information and/or creating reports.

3.2.6 Tax Type Exemptions

Two fields are provided for entering the quantity of tax type exemptions and the pointers to the tax exemption structures.

3.2.6.1 Tax Type Exemptions Field

The Tax Type Exemption indicator specifies the number of tax type exemptions that are included. These are tax type and transaction specific exemptions.

3.2.6.2 Tax Type Exemption Pointer

Tax type exemption(s) are specified with a pointer to one or more tax exempt structures. The user should supply a pointer to the first Tax Type Exemption and all exemptions must be contained in a contiguous block of memory.

3.2.7 Optional Fields

Optional Fields are provided to allow clients to enhance reporting and billing utilities with information beyond the scope of that which is generated to support AFC activities.

The Primary Output Key (POK) is a 20-character text field that is not manipulated during processing and stored as part of the log file record.

It can be used as part of the sorting key (see Figure 3-1) when using some utilities, allowing for the combining of records based upon this and other fields. It is useful when it is desired to have like taxes from different transactions combined, to have taxes from each transaction detailed individually in a report or to have each transaction detailed at the customer level.

Figure 3-1 Primary Output Key

Uniquekey.csv	BillSoft, Inc.-1	0000011+00000166465
	BillSoft, Inc.-1	0000012+00000034549
	BillSoft, Inc.-1	0000013+00000035335
	BillSoft, Inc.-1	0000060+00000094226
	BillSoft, Inc.-1	0000102+00000059900
	BillSoft, Inc.-1	0000131+00000145856
	BillSoft, Inc.-2	0000060+00000000000
	BillSoft, Inc.-3	0000060+00000000000
	BillSoft, Inc.-4	0000060+00000000000
	BillSoft, Inc.-5	0000011+0000038136
	BillSoft, Inc.-5	0000012+0000007915
	BillSoft, Inc.-5	0000013+0000008095
	BillSoft, Inc.-5	0000060+0000021586
	BillSoft, Inc.-5	0000102+0000014391
	BillSoft, Inc.-5	0000180+0000069550
	BillSoft, Inc.-6	0000011+0000011116
	BillSoft, Inc.-6	0000012+0000002307
	BillSoft, Inc.-6	0000013+0000002360
	BillSoft, Inc.-6	0000060+0000006292
	BillSoft, Inc.-6	0000102+0000004000
	BillSoft, Inc.-6	0000131+0000009740
	BillSoft, Inc.-7	0000060+0000009438
	BillSoft, Inc.-7	0000131+0000014610

Primary Output Key

Combined taxes

If you would want like taxes combined at the customer level, then you specify a like Primary Output Key for the records you want combined.

Samekey.csv

Billsoft, Inc.	0000011+00000215718
Billsoft, Inc.	0000012+00000044772
Billsoft, Inc.	0000013+00000045789
Billsoft, Inc.	0000060+00000131543
Billsoft, Inc.	0000102+00000078291
Billsoft, Inc.	0000131+00000170206
Billsoft, Inc.	0000180+00000069550

Primary Output Key

3.3 Company Information

Refer to Table 3-3. The following company information is contained in fields found in the transaction record.

Table 3-3 Company Information	
Description	Valid Entry
Company Identifier	Company Identifier, optional
Facilities Based	Facilities Based (default) = TRUE, Non-Facilities Based = FALSE.
Franchise	Franchise (default) = TRUE, Non-Franchise = FALSE.
Regulated/Unregulated	TRUE = Regulated (default) = TRUE, Unregulated = FALSE.

3.3.1 Company Identifier

The Company Identifier field is an optional field made available for alpha information, such as the name of a subsidiary company. This information is passed thru to the tax log so that reporting utilities can sort or summarize by this field.

3.3.2 Facilities Based Flag

The Facilities Based flag specifies whether the transaction is sold over tangible facilities controlled by the seller. If the seller delivering the service owns or controls the facilities used to provide the service, then the seller is facilities based. If the seller does not own or control the facilities, the seller is non-facilities based. In some jurisdictions, tax outcomes will vary depending on whether the service is delivered over infrastructure controlled by the seller.

3.3.3 Franchise Flag

The Franchise flag indicates that the company provides services sold pursuant to a franchise agreement between the carrier and jurisdiction.

3.3.4 Regulated / Unregulated Flag

The Regulated / Unregulated flag is used to specify if the Telecommunications company and its services are regulated by the regulatory commission in the state of the service.

3.4 Transaction Data

Refer to Table 3-4. The following transaction information is contained in fields found in the transaction record.

Table 3-4 Transaction Data Information	
Description	Valid Entry
Charge	amount charged to customer for transaction
Date	Transaction bill date. This field is provided to allow rating and taxing to occur on a date other than the billing date.
Lines	number of lines (use with transaction type LOCAL and service type LINES)
Minutes	Minutes of call, defaults to zero when not appropriate (NOTE: some taxes are per minute.)
Sale/Resale	TRUE = Sale, FALSE = Resale
Service Type	Refer to Table 2-1 for valid Transaction / Service Type entries
Transaction Type	Refer to Table 2-1 for valid Transaction / Service Type entries

3.4.1 Charge

The Charge field specifies the amount of the transaction to be taxed. This amount will be passed through AFC to rate the tax based on the specified transaction/service pair.

3.4.2 Date

The Date field is normally populated with the bill date. Generally accepted accounting principles dictate that liabilities should be recorded when revenues are recorded. In most cases, neither of these is recorded (or even known) until billing occurs.

However, companies with a high call volume that record revenue daily as it occurs should record the tax on the same basis (i.e. the call date should be used).

AFC compares this date to the effective date of each tax that applies to the transaction. Historical rates and effective dates are maintained and updated within the AFC Engine and AFC will return the correct tax information based upon the transaction date. The monthly updates assure that the rates and effective dates are current.

WARNING:

*All transaction data Date fields must be formatted as CCYYMMDD.
Failure to do so will cause incorrect results when the transaction is processed.*

3.4.3 Lines

When local service is provided, a transaction should be generated with the Lines field populated with the number of lines the customer subscribes to. AFC uses this information for generation of per line taxes usually associated with local E911 charges and local telecommunications relay service taxes and other assorted taxes.

3.4.4 Minutes

The Minutes Field specifies the length of phone call in minutes, with one tenth of a minute precision capability. AFC uses this field for generation of taxes that are specified as per minute flat fees in some taxing jurisdictions and stores the value in the AFC log database.

3.4.5 Sale Type

The Sale Type specifies the whether the transaction is a Retail which is a sale to an end user or Wholesale, resale to another carrier, which will resell the service to an end user or another carrier.

3.4.6 Service Type

Refer to Transaction and Service Types for details of this field.

3.4.7 Transaction Type

Refer to Transaction and Service Types for details of this field.

3.5 Taxing Jurisdiction Identification Information

The tax jurisdiction that can claim nexus for the transaction must be obtained in order for the proper taxes to be applied. Assigning the correct jurisdiction to the transaction is crucial in obtaining and returning the correct tax to the billing system.

Jurisdiction information can be supplied to the AFC Engine in several ways. It can be supplied using a Jurisdiction Code (JCode), Permanent (location) Code (PCode), NPANXXs, Zip Code with city, and state information or FIPS Codes.

3.5.1 The JCode Jurisdiction Identification

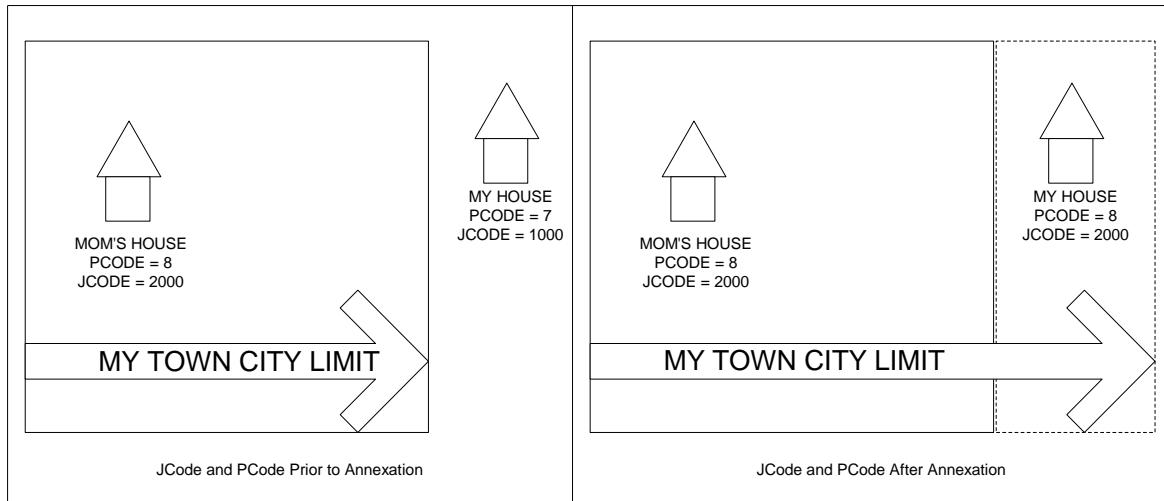
Avalara has assigned each jurisdiction with a proprietary “JCode” (Jurisdiction Code) that identifies its specific territorial boundaries. A jurisdiction fairly claiming nexus on a transaction within this boundary will apply taxes accordingly. The AFC Engine uses this “jurisdiction bound” JCode to associate the jurisdiction specific taxes to the transaction as it is processed.

However, any alterations in the jurisdictional boundaries, such as an expansion of a city limit, will be reflected in the JCode of locations that are situated within the expanded jurisdiction area.

The Avalara tax research staff monitors jurisdiction taxation activity to maintain current and accurate taxation information. The results of this research is applied to the Monthly updates which are provided to ensure the most accurate taxation is applied to the records processed by the AFC Engine. The JCodes will be changed at this time. It is the user’s responsibility to re-populate their database tables with the new JCodes every time an AFC update is provided. If this task is not performed, the AFC Engine will return incorrect tax calculations for jurisdictions that have been updated.

However, when a transaction is submitted with the stable and location sensitive PCode, the AFC Engine can use internal functions to obtain the correct jurisdiction. Since the AFC Engine is updated with the remapped jurisdiction information monthly, tax calculations are always performed with the most current jurisdiction taxation information that applies to the address defined by the PCode.

Figure 3-2 JCode and PCode Changes During Annexation



All that is required to make use of this recommended option is to cross reference or “map” the customer locations with its correct PCode (see Section 3.4.1.1) time invested in incorporating PCodes into your database records will be quickly offset by the reduced processing time and the increased accuracy and reliability of your tax reporting will be evident.

3.5.1.1 Mapping Customer Locations to PCodes

Avalara markets Avalara Geo for Communications to assist in mapping customer locations to PCodes when populating the client customer database. It quickly obtains the current PCode for a specific geographical location.

When performing this task without the aid of EZgeo[•], refer to the all_adr.txt file to look up the PCode that should be used to populate the client database with customer information. This comma delimited ASCII file is supplied for use during AFC setup and when adding new customers to the client database. The file is replaced with a current version when monthly updates are performed (refer to Section 8.6).

all_adr.txt is a comma-delimited text file that contains a cross reference of PCodes to taxing jurisdictions. The format is PCode, Primary / Alternate Location, Country, State, County, Locality, Beginning ZIP Code, Ending ZIP Code. These fields are self-explanatory with the exception of the Primary / Alternate Location (refer to 3.4.1.1).

Primary/Alternate Location

The Primary / Alternate location flag is set to '0' to indicate that the location associated with the PCode is a "Primary" location or '1' to indicate that it is an "Alternate" location.

Secondary locations are original city names that remain in use even though the areas within them have since been divided and given different names.

For example, the city of Overland Park, KS is a primary location. The first post office to serve the area was established at the Shawnee Methodist Mission in the early nineteenth century, the extents of which

encompassed the then non-existent city of Overland Park as well as nearly a dozen other cities (primary locations) which appeared over time. The post office still recognizes the Shawnee Methodist Mission area as Shawnee Mission, KS, and continues to deliver to a primary location when the "Alternate Location" address is provided.

3.5.1.2 Data Entry Modification Procedure

Use the following procedure to acquire the correct information to enter.

1. Have the customer data entry software first request the NPANXX of the customer from the data entry associate.
2. Have the customer data entry software request the zip code of the customer from the data entry associate.
3. Query the NPANXX7 file and the "all_adr" file/database to obtain the associated PCodes.
4. If matching PCodes are found, populate the customer data record with the associated address from the "all_adr" file and request verification from the customer data entry associate.
5. If the PCodes do not match or if the customer data entry associate rejects the automated selection, display all addresses associated with both PCodes and the zip code entered.
6. Allow the customer data entry associate to select the appropriate address from the list.
7. In the unlikely event that the customer address is not displayed, allow the customer data entry person to enter the address supplied by the customer. Store the address with a "special key" as instructed by Avalara Return the address and key to Avalara via E-mail. Avalara will research the missing address and include it in the next monthly update. Avalara will also provide you with a cross reference of the "special key" and the correct "PCode" to allow the customer record to be properly installed.

NOTE:

Avalara subscribes to the United States Postal Database which should preclude this event. Nonetheless, it does occur on occasion and this process has been established for that reason.

Use the PCode from the customer record for the service address when interfacing with AFC.

Resources such as the USPS or the U.S. Census Department web sites can also be helpful in pinpointing a customer location.

AFC functions can also be used to obtain PCodes (see Section 7.8 API Listings). For detailed information about determining the tax jurisdiction which can claim nexus on a transaction, refer to Section 3.5.

3.5.2 NPANXX

NPANXX input is nearly as accurate as the JCode method and performance differences are virtually undetectable. Using this method, the jurisdiction is determined by the three elements of a transaction; the origination, termination and bill-to.

However, there are potential problems associated with using the NPANXX.

1. The first six digits of a toll free 800 or 888 number do not constitute an NPANXX. Rather, these numbers are associated with a “ring to” number which is the number that is actually reached when the 800/888 number is dialed. The NPANXX of the “ring to” number should be used as the NPANXX for the 800 number. This replacement must be made before the information is passed to the AFC system.
2. Switches are capable of producing CDRs with account codes in place of a “Bill To” number, which is not related to an actual NPANXX. The user is required to convert this number to the actual NPANXX using data from the billing system before interfacing with AFC.

NPANXXs exist for the U.S., Barbados, Canada, Guam, Mexico, Northern Mariana Islands, Puerto Rico, Trinidad & Tobago and the US Virgin Islands. These countries are on the North American dialing plan.

Use of NPANXX for determination of a taxing jurisdiction is not sufficient for local service. There are too many foreign exchanges that will incorrectly map customers to the wrong local tax jurisdiction resulting in the wrong local taxes to be applied and unwanted calls to your customer service department.

3.5.3 Zip Codes

This method of providing jurisdiction information is the least efficient method provided. When using the ZIP Code and address information, AFC cannot determine the jurisdiction for a telecommunication transaction since information regarding the origination and termination numbers is not provided. A record submitted using this method will compromise the accuracy of correct jurisdiction identification, the degree of which would depend upon the amount of data provided for the address and the user's ability to select the correct taxing jurisdiction zip code and address. AFC databases necessarily contain numerous duplicate zip codes that cross taxing jurisdiction and locality boundaries. Providing a complete address with zip code will provide the best possible match. If the address information is not provided, AFC will return taxes based upon the first match of the input record information.

If the jurisdiction is positively identified by the billing system it would be appropriate to use the zip code for transactions. For instance, this would apply if the transaction is the sale of a product at a business that has just one location or in the case of internet usage from a residence.

AFC currently has Zip Code information for U.S. and Canadian jurisdictions. Use the Zip Code Plus 4 Interface to use the 6 character Canadian postal codes.

3.5.4 Zip Plus 4

Just as with the Zip Code method (see Section 3.4.3), AFC cannot determine the jurisdiction for a telecommunication transaction since information regarding the origination and termination numbers is not provided. However, it is useful (and more accurate than the 5-digit zip code) because this 9-digit zip code uniquely identifies a location, which can then be used to determine the tax jurisdiction. There is no need to supply the additional city and county information and information in these fields are ignored.

NOTE:

Supplying ZIP Code Plus 4 information where the plus 4 is all zeros will function like the ZIP Code Interface since the United States Postal Service has no valid plus 4 code assigned with all zeros. The default will retrieve the first jurisdiction with the supplied 5-digit zip code.

3.5.5 Canadian Postal Codes

Canadian postal codes need to be specified using the ZIP Plus 4 to retrieve a tax jurisdiction code. Place the first three characters go in the zip code field and the last three characters go in the plus 4 field. In addition, the country ISO field must be set to the Canadian ISO Code.

3.5.6 FIPS Codes

AFC has functions allowing the use of FIPS codes to retrieve tax jurisdiction codes. The US Census Bureau provides the following description of the FIPS Codes and associated reference tables.

“The Federal Information Processing Standards codes (FIPS codes) are a standardized set of numeric or alphabetic codes issued by the National Institute of Standards and Technology (NIST) to ensure uniform identification of geographic entities through all federal government agencies. The entities covered include: states and statistically equivalent entities, counties and statistically equivalent entities, named populated and related location entities (such as, places and county subdivisions), and American Indian and Alaska Native areas”.

A FIPS Code is a 10-digit numeric code with the following format:

Table 3-5 FIPS CODE FORMAT	
SSCCCPFFFF	
SS	FIPS State Code
CCC	FIPS County Code

Table 3-5 FIPS CODE FORMAT	
PPPPP	FIPS Place Code

Figure 3-3 FIPS State Codes for the States and the District of Columbia					
Name	FIPS State Numeric Code	FIPS State Alpha Code	Name	FIPS State Numeric Code	FIPS State Alpha Code
Alabama	1	AL	Montana	30	MT
Alaska	2	AK	Nebraska	31	NE
Arizona	4	AZ	Nevada	32	NV
Arkansas	5	AR	New Hampshire	33	NH
California	6	CA	New Jersey	34	NJ
Colorado	8	CO	New Mexico	35	NM
Connecticut	9	CT	New York	36	NY
Delaware	10	DE	North Carolina	37	NC
District of Columbia	11	DC	North Dakota	38	ND
Florida	12	FL	Ohio	39	OH
Georgia	13	GA	Oklahoma	40	OK
Hawaii	15	HI	Oregon	41	OR
Idaho	16	ID	Pennsylvania	42	PA
Illinois	17	IL	Rhode Island	44	RI
Indiana	18	IN	South Carolina	45	SC
Iowa	19	IA	South Dakota	46	SD
Kansas	20	KS	Tennessee	47	TN
Kentucky	21	KY	Texas	48	TX
Louisiana	22	LA	Utah	49	UT
Maine	23	ME	Vermont	50	VT
Maryland	24	MD	Virginia	51	VA
Massachusetts	25	MA	Washington	53	WA
Michigan	26	MI	West Virginia	54	WV
Minnesota	27	MN	Wisconsin	55	WI
Mississippi	28	MS	Wyoming	56	WY
Missouri	29	MO			

Figure 3-4 FIPS State Codes for the Outlying Areas of the United States, the Freely Associated States and Trust Territory			
Area Name	FIPS State Numeric Code	FIPS State Alpha Code	Status
American Samoa	60	AS	1
Federated States of Micronesia	64	FM	3
Guam	66	GU	1
Marshall Islands	68	MH	3
Northern Mariana Islands	69	MP	1
Palau	70	PW	4
Puerto Rico	72	PR	1
Virgin Islands of the U.S.	78	VI	1

3.5.7 Support for India

Processing tax calculations in the the country of India includes support for all the states. Please reference the table below for the appropriate IDs, abbreviations and PCodes for each state.

States in India					
Ctry Code	ISO	State Id	Abbv	State	PCODE
356	IND	1	AP	Andhra Pradesh	5148401
356	IND	2	AR	Arunachal Pradesh	5148402
356	IND	3	AS	Assam	5148403
356	IND	4	BR	Bihar	5148404
356	IND	5	CT	Chhattisgarh	5148405
356	IND	6	GA	Goa	5148406
356	IND	7	GJ	Gujarat	5148407
356	IND	8	HR	Haryana	5148408
356	IND	9	HP	Himachal Pradesh	5148409
356	IND	10	JK	Jammu and Kashmir	5148410
356	IND	11	JH	Jharkhand	5148411
356	IND	12	KA	Karnataka	5148412
356	IND	13	KL	Kerala	5148413
356	IND	14	MP	Madhya Pradesh	5148414
356	IND	15	MH	Maharashtra	5148415
356	IND	16	MN	Manipur	5148416
356	IND	17	ML	Meghalaya	5148417
356	IND	18	MZ	Mizoram	5148418
356	IND	19	NL	Nagaland	5148419
356	IND	20	OR	Odisha	5148420
356	IND	21	PB	Punjab	5148421
356	IND	22	RJ	Rajasthan	5148422

States in India					
Ctry Code	ISO	State Id	Abbv	State	PCODE
356	IND	23	SK	Sikkim	5148423
356	IND	24	TN	Tamil Nadu	5148424
356	IND	25	TG	Telangana	5148425
356	IND	26	TR	Tripura	5148426
356	IND	27	UP	Uttar Pradesh	5148427
356	IND	28	UT	Uttarakhand	5148428
356	IND	29	WB	West Bengal	5148429
356	IND	30	AN	Andaman and Nicobar	5148430
356	IND	31	CH	Chandigarh	5148431
356	IND	32	DN	Dadra Nagar Haveli	5148432
356	IND	33	DD	Daman and Diu	5148433
356	IND	34	DL	Delhi	5148434
356	IND	35	LD	Lakshadweep	5148435
356	IND	36	PY	Puducherry	5148436

3.6 Jurisdiction Identification Details

Obtaining the correct user's location is critical in calculating the local taxation.

3.6.1 Determining the Taxing Jurisdiction for Wireline

AFC applies taxes to transactions based on the statutes that dictate specific taxes per jurisdiction. The AFC engine contains current taxing information for jurisdictions, but for AFC to calculate taxes for long distance telecommunication calls correctly it must determine the taxing jurisdiction for the transaction based on the following:

- Origination Jurisdiction – The Origination location of the transaction to be taxed. For telecom activity, this is usually specified by the number called from, also known as the “From Number” or “Origination number.”

- Termination – The Termination location of the transaction to be taxed. For telecom activity, this is usually specified by the number called, also known as the "To Number" or the "Termination number."
- Service Address – The Service Address of the transaction to be taxed. For telecom activity, this is usually specified by the service location; also known as the Bill To Number or BTN.

Each jurisdiction from the state level down to the local level establishes jurisdiction rules to determine if their tax will apply to a specific call transaction. These jurisdiction rules (also known as sourcing rules) for determining the correct jurisdiction arose from a United States Supreme Court ruling that established the minimum requirements necessary for a taxing jurisdiction to claim “nexus” and validly tax a particular wireline long distance telecommunication transaction.

Known as the Goldberg Rule, or the “2 out of 3 Rule,” the ruling requires that two out of the three sourcing elements (origination, termination, and service address) must take place in a particular jurisdiction in order for that jurisdiction to tax the transaction. Figure 3-6 provides two examples to illustrate this principle.

Figure 3-5 Correct Jurisdiction Transaction Taxation applying Goldberg Rule



The first example is a call that originates in Oregon, terminates in California and has a service address (BTN) in Oregon. Since “2 out of 3” sourcing elements take place in Oregon, Oregon may tax this transaction.

The second example is a call that originates in New Mexico, terminates in Texas and has a service address (BTN) in Kansas. Since no state has more than 1 out of the 3 sourcing elements, the transaction is not taxable by any state.

The “Goldberg Rule” or “2 out of 3” is the minimum standard. Many jurisdictions apply rules that apply stricter nexus requirements. For example, a jurisdiction might require that a transaction both originate and be billed in that jurisdiction in order to be taxed. This is a stricter requirement since the jurisdiction has eliminated one of the elements as a possibility. Fewer transactions will be taxable by this rule than by the straight “Goldberg Rule”. No jurisdiction may impose a rule that will result in more transactions being taxable than the “Goldberg Rule”.

3.6.1.1 Incorrect Jurisdiction Assignment

A customer may be assigned an incorrect jurisdiction if the geographical information is not thoroughly researched. In the example shown in Figure 3-7, the customer (marked by an X) resides in a rural area located in Pawnee County, Nebraska, just outside the city limits of Summerfield, Kansas.

Figure 3-6 Geographical Anomaly



However, in all likelihood this customer's phone line is served by a switch in Summerfield, Kansas and will therefore have a Summerfield, Kansas NPANXX. It is also likely that the customer's mail service is provided by the Summerfield, Kansas Post Office and therefore has a Summerfield, Kansas mailing address.

This geographical anomaly will cause an incorrect assessment of taxation if not researched properly. If the AFC user enters the customers NPANXX and Service Address information into the transaction records, the customer will be incorrectly assessed taxes according to those of the Summerfield, Kansas jurisdiction instead of those for the unincorporated area of Pawnee County, Nebraska jurisdiction.

3.6.1.2 Tax Situsing

Although the jurisdiction is determined by the "2 out of 3 rule" for the majority of long distance telecommunications taxes, it is only one of 10 rules specified for jurisdiction determination by taxing jurisdictions in the United States.

1. Tax based on 2 out of 3 Rule – Apply tax where 2 out 3 jurisdictions are equal. This is usually the default option.
2. Tax based upon Billing – Apply tax based on the billing location.
3. Tax based upon Origination – Apply tax based on the Origination of the call location.
4. Tax based upon Termination – Apply tax based on the Termination of the call location.
5. Tax if (Bill=Orig) – Apply tax only if billing jurisdiction and origination jurisdiction is equal.
6. Tax if (Bill=Orig=Term) – Apply tax only if all three jurisdictions are equal.

7. Tax if (Bill=Tax Jurisdiction) – Apply tax if billing jurisdiction equals origination or termination jurisdictions.
8. Tax if (Bill=Term) – Apply tax only if billing jurisdiction and termination jurisdiction is equal.
9. Tax if (Orig=Tax Jurisdiction) – Apply tax if origination jurisdiction equals billing or termination jurisdictions.
10. Tax if (Term=Tax Jurisdiction) – Apply tax if termination jurisdiction equals billing or origination jurisdictions.

3.6.2 Determining the Taxing Jurisdiction for Cellular

For cellular transactions, the service address (BTN) is used to determine the jurisdiction. Cellular services by their inherent nature are nomadic, making the origination point of a phone call transient and prone to difficulty in tracking. In an effort to resolve this uncertainty, Congress passed the Mobile Telecommunications Sourcing Act (MTSA). The MTSA is a federal law which requires all state and local taxes to adopt language sourcing cellular services to the customer's "place of primary use" (PPU) or the location where the customer predominately uses their cell phone. The federal statute allows the seller to presume that the place of primary use is the customer's billing address until the customer notifies them that this is not the case.

3.6.3 Determining the Taxing Jurisdiction for VoIP

VoIP services may be utilized on a nomadic basis with difficulties in determining the originating location at the time of the call. Unfortunately, Congress has not passed a Primary Place of Use (PPU) statute for VoIP; however, the same guidelines outlined in the Mobile Telecommunications Sourcing Act (MTSA) are the generally followed industry practice. The seller is allowed to presume the PPU based on the customer's billing address until the customer notifies them that this is not the case.

3.6.4 Determining the Taxing Jurisdiction within Canada

Sourcing telecommunication taxes in Canada differs in that determining the "place of supply" must be clearly identified through application of the following rules:

1. A supply of a telecommunication service, which produces the availability of telecommunications facilities, (other than a service of granting sole access to a telecommunications channel) is made in a province if:
 - all of the facilities are ordinarily located in that province; or
 - where not all of the facilities are ordinarily located in the province, the invoice for the supply is sent to an address in that province.

2. For other types of supplies of telecommunication services (other than a service of granting sole access to a telecommunications channel), the supply is made in a province if the telecommunication:
- is both emitted (originating) and received (terminating) in that province;
 - is either emitted (originating) or received (terminating) in that province and the billing location for the service is located in that province; or
 - is emitted (originating) in the province and received (terminating) outside the province and the billing location for the service is not in a province where the telecommunication is emitted or received.

4. AFC Calculations

The AFC Engine accepts the transaction record, processes the information provided and creates a table containing the generated tax information. Additional methods to process specialized taxes are available through the use of AFC functions.

4.1 Meeting the Requirements of Specific Tax Issues

Many tax issues are resolved within the AFC Engine, thereby relieving the user from making changes to account for them. Additional functions are provided to meet other special taxation issues.

Table 4-1 Specific Tax Issues

Category	Description
Specific Tax Exemptions	Specifies tax type at specific tax level for exemption
Tax Adjustments	Allow for adjustment activities such as refunds, changing a customer's bill or when terminating un-collectable accounts.
Tax Overrides	Allows for a change of a tax rate.
Tax Rate Brackets	Tax rate that changes as the taxable amount of the transaction increases.
Rate at Final	A specialized case of tax brackets.
Discount Adjustment	Allows for entry of different discount adjustment types.
Tier on Transaction	Allows for tax to be determined using graduated tax brackets.
Tax on Tax Until no Effect	Option to control calculation of tax on tax
Taxed Taxes	Allows for processing when one tax includes in its base the tax calculated from another tax.
Get Rates	Retrieves current tax information in a form suitable for overrides.

4.1.1 Tax Type Exemptions

Tax Type Exemptions are used to specify a specific Tax Type at a specific Tax Level to be exempted for the current transaction. The exemption jurisdiction code specifies the jurisdiction for the tax exemption. If the jurisdiction code is not specified (i.e. set to zero), then all taxes of the Tax Type and Tax Level specified are considered exempt regardless of the jurisdiction they are calculated for. Typically the JCode should be specified as specific tax exemptions are normally only effective for specific jurisdictions.

Another option allows the tax type to be set to zero, to indicate that all taxes of a specific tax level are exempt in the specific jurisdiction.

4.1.1.1 Declaring Tax Type Exemptions within a Transaction Record

When setting up the AFC transaction record, there are fields that specify either exempt at a tax level or a specific tax. For more information, reference Sections 3.1.4 for exemption levels, 3.1.7 for Jcode exemption levels and 3.1.10 for tax type exemption levels.

4.1.1.2 Declaring Tax Type Exemptions Using Functions

Alternatively, Tax Type Exemptions can be applied using functions (such as APIs). The following fields are required for this.

Table 4-2 Tax Exempt Record

Tax Type	Tax Type identifier
Tax Level	Tax Level identifier
JCode for Exemption	Jurisdiction for exemption

In addition, the transaction contains a pointer set to the tax exempt to exempt a specific tax.

Table 4-3 Tax Exemption Fields

tax_exempt	Identifies the number of exemption records
*s_exempt	Pointer to the Tax Exempt record (see Table 2-4) or Array of Tax Exempt records. If no records are setup, this value should be null.

4.1.1.3 General Tips When Declaring Tax Exemptions

1. The JCode is used to specify tax exemptions because exemptions are normally only effective for specific jurisdictions.
2. Federal taxes cannot be exempted using a level exemption, but can be exempted using specific exemptions. Some taxes, such as the Federal USF tax, are non-exemptible using level exemptions. These non-exemptible federal taxes may also be overridden to exempt them. Refer to Section 4.1.3 for more information on Tax Overrides.
3. It is easy to evaluate the exemptions passed through AFC because they are tracked separately and stored in the tax log by type of exemption.

4.1.2 Tax Adjustments

Tax adjustment functions provide for adjustment activities such as refunds, changing a customer's bill or writing off un-collectable accounts. The AFC engine allows adjustments to be passed by the user. Note that these adjustments must be accounted for to provide an audit trail.

Tax adjustments are made using the Adjustment functions. The tax amount entered should be a positive number for credit adjustments as the adjustment functions will appropriately sign the charges, lines or locations for computation purposes. The adjustments are logged in the EZTax.log along with the rest of the tax and transaction data associated with the tax run. The information is returned to the billing system and is utilized to update tax data for report generation and compliance filing.

Adjustments may be declared as 'default' in which case they are processed exactly like a similar charge transaction with negative tax results. They may be declared as least or most favorable. ***This declaration is only useful for taxes which have multiple tiers or brackets and should be used with caution.*** The rate will be determined separately for each tax returned within the tax table. AFC software will search the tiers or brackets and select the tier which has either the highest rate (for most favorable) or lowest rate (for least favorable) and apply that rate to the number of lines or charge amount as appropriate.

4.1.2.1 General Tips When Making Tax Adjustments

1. To calculate adjustments accurately, AFC requires the following activity:
 - a. Call AFC using an adjustment API. (Please refer to the APIs provided in Section 7.8 API Listings).
 - b. Send positive values for change and lines.
 - c. Set the adjustment method (see the Adjustment Method Table below).
 - d. Set the discount type (see the Discount Type Table in Section 4.1.7).
2. If logging is turned on, adjustments are stored in the eztax.log file with the other transactions.

Adjustment Method Table	
Name	Description
Default	Tax brackets applied normally.
LeastFavorableRate	Tax brackets applied to produce smallest tax refund.
MostFavorableRate	Tax brackets applied to produce largest tax refund.

4.1.2.2 Line Based Adjustments

Normal adjustments are managed with the processes outlined throughout this section; however, there are exceptions to this process when managing line based adjustments in Invoice Mode. When adjustments are made in Invoice Mode with line based transactions, AFC will use the rate for the total number of lines within the adjustment(s). The system will not make any assumptions regarding the purchase rate of the original lines. Therefore, the rate charged will reflect the total number of lines received. (Please see the example provided in the scenario below).

Adjustment Scenario		
Number of Lines	Rate	Total Tax Amount
Customer Orders 1 line	Rate @ 0.67	Tax = 0.67
Customer Orders 5 lines	Rate adjusted to 6 lines @ 0.34	Tax+=1.37
Customer Orders 40 lines	Rate adjusted to 46 lines @ 0.22	Tax+=8.08
Customer Orders 5 lines	Rate adjusted to 51 lines @ 0.17	Tax-=1.45

Customer adjustment 7 lines: rate @ 0.34 tax =2.38

Rate Table			
Lines	Tax	Lines	Tax
1-10	5.92	81-90	53.28
11-20	11.84	91-100	59.20
21-30	17.76	101-110	65.12
31-40	23.68	111-120	71.04
41-50	29.60	121-130	76.96
51-60	35.52	131-140	88.80
61-70	41.44	141-150	59.20
71-80	47.36	151-160	94.72

4.1.3 Tax Overrides

Overrides allow the client to change the rate of a tax in the AFC Engine. Avalara markets the AFC Manager – Rate and Logic Modifier (a Graphic User Interface based Windows program that is sold separately) to support this activity. It steps the user through the process of creating an EZTax.ovr file. Alternatively, overrides can be achieved using the Override functions (such as APIs).

WARNING: An override to exempt taxes **OVERRIDES** the tax information in Avalara's tax research database. This is not recommended for those that do not possess a full understanding of the tax ramifications and liabilities when doing so.

Although all clients can use Tax Overrides, the method to make use of them depends upon the client type.

1. AFC SaaS Standard clients must use the AFC Manager – Rate and Logic Modifier to create an EZTax.ovr file that contains the overrides to insert. The file is then uploaded to the service bureau inside the FTP.zip file that carries the CDS file for processing. The overrides are then used during that processing and for subsequent processing until a new or empty override file is uploaded.
2. Avalara clients that submit records in Batch fashion must use the AFC Manager - Rate and Logic Modifier to create an EZTax.ovr file that contains the overrides to insert. Filelocs.txt is then

modified to point to this file. When the data file is processed, the overrides are inserted automatically.

3. Avalara clients that integrate with the AFC Engine using functions (such as APIs) can use the Override utility as an easy way to create overrides in the same manner that batch mode and online clients do, modify filelocs.txt to point to the EZTax.ovr file or use the file path to point to the EZTax.ovr file.

The following functions use a session and a tax override record to override the tax rates used by AFC to calculate taxes.

Table 4-4 Tax Override Names and Descriptions	
Function Name	Override Description
EZTaxOvrJCodeEx** **This is the override JCode function for the most current version of AFC.	Performs tax override using a JCode to designate the Jurisdiction.
EZTaxOvrPCodeEx	Performs tax override using a PCode to designate the Jurisdiction.
EZTaxOvrZipEx	Performs tax override using zip code and address information to designate the Jurisdiction.

4.1.3.1 Tax Override Fields

AFC functions can be used to override a specific tax rate used by AFC.

Table 4-5 Tax Override Fields	
Field	Description
Scope	Scope of override
Type	Tax type identifier
Level	Tax level identifier
Exempt Level	Level of Exemption
Limit	Maximum lines or charge to apply tax to
Date	Effective date of Tax Rate
Tax Rate	Tax Rate
Previous Tax Rate	Previous tax rate
Maximum Base	Maximum amount to apply tax
Excess Tax	Rate for amount above the Maximum Base
State Override	Override state rate if present
County Override	Override county tax if present
Replace State Tax	Tax replaces state tax
Replace County Tax	Tax replaces county tax

Scope

Refer to Table 4-6. The term Scope is used to indicate the magnitude of the override.

Table 4-6 Tax Level Effect of Exemption

Tax Level	Scope	Effect of Exemption
Federal	Federal	Exempt all defined Federal taxes
Federal	State	Exempt entire State for the specific Federal taxes
Federal	County	Exempt the Federal taxes within a specified County
Federal	Locality	Exempt the Federal taxes within a specified Locality
State	State	Exempt all Defined State taxes within a State
State	County	Exempt the State taxes within a specified County
State	Locality	Exempt the State taxes within a specified Locality
County	State	Exempt all County taxes within a specified State
County	County	Exempt all County taxes within a specified County
County	Locality	Exempt the County taxes within a specified Locality
Locality	State	Exempt all Locality taxes within a specified State
Locality	County	Exempt all Locality taxes within a specified County
Locality	Locality	Exempt all Locality taxes within a specified Locality

As an example, specifying a gross receipts tax at the Local level with a scope of State level will cause all jurisdictions in the state identified by the jurisdiction to be passed to the AFC Engine with the specified tax overridden. The County level affects all taxes in the County and Local level affects only the jurisdiction specified. The specification of Federal level is only valid with federal taxes which will always be overridden at the Federal level.

Type

The Tax type identifier is used to define the type of tax for which the override will apply. Refer to Table 4-19 for a complete list of Tax Types supported in AFC.

Level

The Tax level identifier is used to define the level for which the override will apply.

Exempt Level

Refer to Table 4-6. The Exempt Level indicator is used to define whether a tax can be exempted by an exemption for all taxes at the same level as this tax. TRUE indicates that the tax can be exempted while FALSE indicates that it cannot be exempted. Note that the tax can still be exempted by a specific tax exemption.

For instance, a state level universal service fund will be exempt if this field is set TRUE and a state level tax exemption flag is passed to AFC. If the flag is set TRUE an exemption at the level of the tax will exempt the tax. If the flag is set FALSE an exemption at the level of the tax will have no effect on that specific tax.

[Limit](#)

The limit indicator is only used for taxes applied per line. When this field is set to zero it indicates no limits are in effect for the specified tax. When the limit is not zero, AFC will apply the tax based upon the number of lines up to, but never exceeding, the limit amount. This has no effect on sales taxes

[Date](#)

The Date field is used to define the effective date that the tax rate is active. The transaction date is compared to the effective date to determine if the current tax rate or the previous tax rate is to be applied.

[Tax Rate](#)

If the taxes are computed as a percentage of the charge then the Tax Rate field is used to indicate the tax rate. For example, if a 5% sales tax were applied then the tax rate would be entered as .05. For all other taxes such as per line, fixed, and per minute, the dollar amount should be entered for the tax.

[Previous Tax Rate](#)

The Previous Tax Rate field is supplied for use when a previous tax is to be used for the tax rate.

[Maximum Base](#)

The Maximum Base defines the maximum charge that the tax is applied to. Any charge above the maximum base is charged at the excess tax rate.

[Excess Tax](#)

If the tax only caps the charge that the tax is applied to then set the excess tax to zero.

[State Override](#)

The Locality or County sales tax record can override the state sales tax rate. This is useful where localities or counties have a special agreement with the state to collect the state tax at a different rate. If the tax is zero then no override will be in effect.

[County Override](#)

The Locality sales tax record can override a County sales tax rate. This is useful where localities have a special agreement with the County to collect the County tax at a different rate. If the tax is zero then no override will be in effect.

Replace State Tax

This option is made available for the rare occasion when a Locality or County sales tax replaces the state sales tax completely.

Replace County Tax

This option is made available for the rare occasion when a Locality sales tax replaces a county sales tax completely.

4.1.3.2 Tax Override Options

The Enhanced Override is used to override taxes.

Table 4-7 Tax Override Options

Name	Description
Enhanced Override	Required fields for the Enhanced Override.
Enhanced Date Override	Required fields for Date Override.
Enhanced Rate Override	Required fields for Rate Override.

Enhanced Override

The Enhanced Override requires the following fields.

Table 4-8 Enhanced Override Fields

Scope	Scope of override
Type	Tax Type identifier
Level	Tax Level identifier
Date Count	Number of date records (normally 2)
Date Table	Address of array of date records

Enhanced Date Override

The Enhanced Date override requires the following fields.

Table 4-9 Enhanced Date Override Fields

Override Date	Starting (effective) date for this set of tax rates
Rate Count	Number of rate records for this date (normally 1)
Level Exempt	Indicates if tax can be exempted by an exemption for all taxes at the same level as this tax. Tax can still be exempted by specific tax exemption.
Rate Table	address of array of rate records

Enhanced Rate Override

The Enhanced Rate Override requires the following fields and rate entries in rate table for taxes.

Table 4-10 Sales Rate Override Fields

Tax	Tax Amount (rate)
Maximum Base	Max amount subject to this tax(end of bracket)

Table 4-10 Sales Rate Override Fields

Replace State	Tax replaces the state tax
State Override	Overrides the state rate if present
Replacement County	Tax replaces the county tax
County Override	Overrides the county tax if present

4.1.3.3 Get Rates

The AFC Get Rates function can be used to build the current tax information into the override structure. Specific changes can be made to returned tax entries to create custom overrides.

Table 4-11 Get Rates Override Fields

PCode	Jurisdiction PCode
Tax Count	Count of all taxes for this jurisdiction
*Taxes Table	Table of all taxes for this jurisdiction in override format

NOTE:

The scope value will be set to the tax level.

4.1.3.4 General Tips When Using Overrides

1. Overrides are not permanent. When an AFC session is exited the created overrides are removed and no longer available. They can also be removed without exiting a session using the EZTaxRestore function.
2. AFC cannot override a tax that does not exist. If an existing tax is a rate, it is necessary to override the tax as a rate, not a per line or fixed amount. Failure to do so will result in incorrect and high tax amounts.
3. Non-exemptible taxes may be overridden to exempt taxes that are normally not exemptible.
4. The AFC Override functions can be used to override the rates, but they will not affect the rules that
5. Determine what taxes are applied to different transaction/service pairs.

4.1.3.5 General Rules for Configuring Override File

1. The override file path must be listed in filelocs.txt.
2. The override file must be the last item in filelocs.txt.

3. There cannot be any blank lines in filelocs.txt.

Also, in order to verify that an override file is in the process of being loaded, the user must go to the AFC Directory and locate the .sta or status file which indicates that an override file has been successfully loaded.

4.1.4 Tax Rate Brackets

Some jurisdictions will dictate a tax rate that changes as the taxable amount of the transaction increases. These break points at which the changes occur define the brackets (or steps) and are most commonly based on dollar amount ranges although other units of measure exist. The rate may increase or decrease according to usage levels.

AFC supports these transactions with an unlimited number of tax brackets. The Avalara Tax Research department continually researches jurisdictions for specific tax practices, such as tax rate brackets, updating the AFC Engine monthly. These updates occur automatically and the user is not required to make changes to account for this.

An example of this is illustrated in Figure 4-1. If a jurisdiction has a general sales tax set at 2% for the first \$500 of a single transaction and set at 1% for that which is over \$500, the tax for a \$1200 sale would be calculated as shown in the figure.

Figure 4-1 Example of Sale with Tax Brackets						
First \$500 of Sale	\$500	@	2%	=	\$10.00	
Remaining Amount of Sale (Over \$500)	\$700	@	1%	=	7.00	
					\$17.00	Total Tax

4.1.4.1 Rate at Final

Rate at Final is a variation of the typical tax bracket. Basically a lookup table, each bracket is defined by the break points at which a rate change occurs and contains the rate to be charged for the total transaction amount falling within the range.

For example, suppose a state establishes an E911 tax with a tax rate dependent on the total number of lines as shown in Figure 4-2.

Figure 4-2 Example of Tax Brackets	
Lines	Rate
1	\$0.19
2	\$0.14
3	\$0.13
4 to 5	\$0.11
6 to 10	\$0.10
11 to 25	\$0.08
26 to 50	\$0.06
51 to 99	\$0.05

Figure 4-2 Example of Tax Brackets	
Lines	Rate
100+	\$0.04

If a customer has 7 lines, the tax rate of \$0.10 per line will apply for a total tax of \$0.70.
If a customer has 2 lines, the tax rate of \$0.14 per line will apply for a total tax of \$0.28.

4.1.4.2 Limits

Some jurisdictions have established tax rates that either take effect or cease to take effect at a specific threshold, defined as a currency value. The point at which this occurs is referred to as a cap or limit. AFC supports these transactions and the user is not required to make changes to account for it.

As an example, if a jurisdiction charges a 10% UTT on only the first \$10 of an invoice, the tax for a \$20 invoice would “cap” at the \$10 threshold, resulting in a ($\$10 \times 10\% = \1) UTT fee.

As an example of the converse, if a jurisdiction does NOT tax the first \$25 of Internet Access usage, a \$35 charge would be reduced by the \$25 threshold “limit,” resulting in a ($\$35 - \$25 = \$10$) taxed amount.

4.1.5 Surcharges

When a government entity levies a surcharge, AFC automatically includes this surcharge in the base of the FET. The most common occurrence of this is found in E911 surcharges. The returned tax table shows which tax entries are considered surcharges.

4.1.6 Prorated Taxes

The AFC Prorating functions send in a percentage that is used to calculate taxable amount on the rated tax and calculates a percentage of the line and fixed amounts. These functions check the prorated logic of each tax in the appropriate jurisdiction to determine if prorating is allowed. If prorating is not allowed, the full amount is taxed.

4.1.7 Discount Adjustment

AFC has an additional table that stores discount types by state with an “allow ability” indicator. The adjustment functions have arguments for the discount type which look up the discount type from the table to determine whether to apply taxes or not.

Discounts may or may not be taxed within each state. When a discount is taxed, the customer receives a tax benefit commensurate with the amount of the discount (i.e., if the customer gets \$5 off on a transaction subject to a 5% tax, the customer pays \$0.25 less in tax than they would have). When a

discount is not taxed, the customer receives no tax benefit from the discount. Whether a discount is taxed or not depends on the type of discount and the rules in a particular tax jurisdiction.

AFC provides five discount types to use in defining the particular discount to be applied and are fully described with supporting example in the following sub-sections.

Discount Type	
Name	Description
None	Discount Type not applicable.
RetailProduct	An amount subtracted from the original price to arrive at a lower price.
ManufacturerProduct	A credit applied to the total amount reimbursed to either the retailer or the customer by the manufacturer.
AccountLevel	A stand-alone discount that is not applied against any service but instead as a stand-alone product.
Subsidized	A credit for telephone service where the telephone provider provides a service to a lifeline eligible customer. The credit will be applied to the subscriber line charge.
Goodwill	A credit applied to customer invoices for the purpose of engendering customer goodwill. For example, compensation for a service outage.

4.1.7.1 Account Level

Especially prevalent in large accounts, this discount is a stand-alone discount that is not applied against any service but instead as a stand-alone product.

Example Description of Account Level

A Kansas customer spends significant amounts on a wide range of services including local exchange, intrastate toll, and interstate toll. The customer's purchasing activity occurs in several states and across multiple accounts. The customer's spending levels earn it a \$1,000.00 discount that is not applied to any particular product or service, but will be applied at an account level.

XYZ Phone Company's Tax Department has determined that account level discounts in Kansas will receive full tax credit, so the discount is mapped to a tax category created to represent generic account level discounts and the customer gets a \$53.00 credit of Kansas state sales tax along with the \$1,000.00 discount. The application of Kansas sales tax to the discount, rather than other state's sales taxes (i.e., states in which there was purchasing activity that helped to earn the discount), is a consequence of the discount being applied to a Kansas account and The Tax Department has determined that the related tax risk is acceptable.

[4.1.7.2 Goodwill](#)

A credit applied to customer invoices for the purpose of engendering customer goodwill. For example, compensation for a service outage.

Example Description of Goodwill

A Kansas customer buys a second line and gets voice mail free for a month (\$6.00 value). The free voice mail is a Goodwill Discount because, although it is offered by the retailer and applied to a particular product or service, the terms of the promotion provide that the discount will not be taxed.

To accomplish this, the billing system will make separate calls to the taxing engine of \$6.00 for the monthly recurring voice mail charge and -\$6.00 for the Goodwill Discount. Both transactions will be represented by the same tax category, but the billing system will send an additional value on the discount transaction indicating that it is a Goodwill Discount. The \$6.00 charge for voice mail generates \$0.32 in Kansas state sales tax. When the -\$6.00 discount is processed for tax applications, the taxing engine determines that a Goodwill Discount is not taxed and generates -\$0.00 in Kansas state sales tax. The offsetting tax amounts are presumably netted together in the tax summary on the customer's bill. Whether the charge and discount amounts are netted on the customer's bill is up to the billing system, and does not affect the tax calculation or the presentation of tax on the bill.

[4.1.7.3 Manufacturer Product](#)

A credit applied to the total amount reimbursed to either the retailer or the customer by the manufacturer.

Example Description of Manufacturer Product

A Kansas customer buys a satellite dish for \$300.00 and receives a \$50.00 rebate (discount) from the satellite company. The billing system will make separate calls to the taxing engine for the dish charge of \$300.00 and the discount of -\$50.00. Both transactions will be represented by the same tax category, but the billing system will send an additional value on the discount transaction indicating that it is a Manufacturer Discount. The \$300.00 charge for the satellite dish generates \$15.90 in Kansas state sales tax. For the -\$50.00 discount, the tax engine determines that Kansas does not allow any tax credit on Manufacturer Discounts and generates \$0.00 in Kansas state sales tax. The offsetting tax amounts are presumably netted together in the tax summary on the customer's bill. Whether the charge and discount amounts are netted on the customer's bill is up to the billing system, and does not affect the tax calculation or the presentation of tax on the bill.

[4.1.7.4 Retail Product](#)

A retail product discount is an amount subtracted from the original price to arrive at a lower price.

Example Description of Retail Product

A Kansas customer has voice mail and is charged a monthly recurring charge of \$6.00/month. The customer buys a second line and gets voice mail free for a month. The billing system will make separate calls to the taxing engine for the monthly recurring charge of \$6.00 and the discount of -\$6.00. Both transactions will be represented by the same tax category, but the billing system will send an additional value on the discount transaction indicating that it is a Retailer Discount.

The \$6.00 charge for voice mail generates \$0.32 in Kansas state sales tax. For the -\$6.00 discount, the tax engine determines that Kansas allows a full tax credit on Retailer Discounts and generates -\$0.32 in Kansas state sales tax. The offsetting tax amounts are presumably netted together in the tax summary on the customer's bill. Whether the charge and discount amounts are netted on the customer's bill is up to the billing system, and does not affect the tax calculation or the presentation of tax on the bill.

[4.1.7.5 Subsidized](#)

A credit for telephone service where the telephone provider provides a service to a lifeline eligible customer. The credit will be applied to the subscriber line charge.

Example Description of Subsidized

A Kansas Lifeline customer purchases local exchange service. Local exchange service is normally \$24.00, but Lifeline customers are charged \$20.50 and the balance of \$3.50 is drawn from a federal government fund for the subsidization of local exchange service to Lifeline customers. The company still has \$24.00 in revenue and will owe Kansas state sales tax on the entire \$24.00 in revenue. The company is prohibited from drawing on the federal government fund to pay for the tax on the \$3.50, so the customer must pay all of the applicable tax.

[4.1.8 Tier at Transactions](#)

In some states, the sales tax on a product has tier taxing logic. When tax is calculated using Tier at Transactions the tax is determined using graduated tax brackets on each transaction separately rather than the amount of the customer invoice. The tax applies tax brackets based on the single product transaction. Tier at Transaction is not the default setting in AFC.

[4.1.9 Tax on Tax Until no Effect](#)

The default application of tax on tax is to make one pass through the calculation of each tax, adding the appropriate taxes to the base amount to be taxed. If the tax on tax until no effect option is selected, AFC will continue to recalculate tax on tax until the amount added is less than 0.005 (one half of a cent).

[4.1.10 Historical Tax Rates](#)

AFC maintains an unlimited number of past tax rates. The tax rate history tables contain the tax rates and effective dates and are referenced when tax calculations are performed on date other than the current date.

[4.1.11 Taxed Taxes](#)

Taxed taxes are instances in which one tax includes in its base the tax calculated from another tax. This can happen when the taxed tax was passed on from the carrier to the end user, thus being revenue to the carrier. This commonly happens with the Federal USF, but it can happen with other taxes as well. As the scope of this document prohibits a discussion of the many examples of taxed taxes, the most common occurrence is discussed here with an example to serve as an introduction to the topic.

AFC applies Federal excise tax to surcharges, the federal universal service funds and the Federal telecommunications relay service fund. The taxation of the federal universal service funds is performed because they are not taxes by definition. In addition, both the federal universal service funds and the federal telecommunications relay service fund are levied against telecommunications companies, not end users.

Telecommunications companies are allowed, but not required, to pass these fees on to their customer base. As these fees are revenue, not taxes, collecting revenue based upon a provided service is taxable as payment for that service.

When the federal excise tax is multiplied by the amount passed to AFC for a transaction and the tax generated by AFC doesn't agree with the calculation it is due to this situation. This phenomenon is not limited to the federal excise tax. It is, however, where it is most noticed due to the volume of federal excise taxes and the common knowledge of its rate.

Refer to Figure 4-3. In the case where the Federal USF and the FET are returned for a transaction, the Federal USF is automatically included in the base of the FET. If both taxes are returned for a \$100 charge, the taxes would be calculated as follows:

Figure 4-3 Taxed Taxes Example (tax rate of 3% is used in this example)					
Tax A			FET		
Charge		\$100.00	Charge		\$100.00
Rate	x	3 %			
Tax Amount	=	\$3.00	→ Tax A	+	\$3.00
			Tax B base	=	\$103.00
			Rate	x	3%
			Tax Amount	=	\$3.09

4.2 Tax Logging

Once the tax calculations are complete, the EZTax.log is created. The log is a binary database containing all of the taxes generated during the tax run. Information in the log includes the jurisdictional data, tax types, tax levels, sale amounts, tax rates, tax amounts, exemptions, adjustments, customer number, etc.

Once AFC performs a graceful exit, the EZTax.log is available for use. This file is commonly used to create tax returns, provide audit trails and internal tracking, and used when making projections.

The options available for reporting the tax information vary based on the integration method that the client has chosen to use.

1. Batch mode clients – The EZTax.log is automatically produced when processing the .CDS file. Several utilities are available for use to produce files to be imported into the billing system. Refer to Section 5.1 for utility selection assistance.
2. AFC SaaS Standard clients - The EZTax.log is automatically produced and the sorting and reporting activities are performed automatically.
3. On-site clients - The EZTax.log is automatically produced. Many sorting and reporting utilities are available for use to produce the desired output file. Refer to Section 5.1 for utility selection assistance.
4. APIs - Clients who use the functions (such as APIs) must turn logging on to gain the benefit of sorting utilities. All of the information used for billing is returned real time during taxation.

4.3 Returned Taxes

Once the tax calculations are completed, the tax information is placed in the Tax Table, available to be returned to the user from the AFC engine. An additional Tax Table will be returned if using Invoice Mode.

4.3.1 Taxes Table

The AFC Table is dynamically allocated during AFC session initialization. This table is used to store tax information as AFC processes transaction records. The size of this table is dependent upon the maximum taxes that can be generated for a single transaction. As such, the size of this table can change from month to month as new taxes are generated or removed from taxing jurisdictions. It is always safe to access from 0 (zero) to [tax_count_returned -1] locations of the table. The user is cautioned to treat this as a read only area. Attempting to access locations that do not exist will result in access violations on most operating systems.

Each function that performs tax calculations or tax exceptions returns a count of taxes generated for the specified transaction. The transaction tax data can be retrieved for use in a billing system by accessing the Tax Table pointer which indicates the location of the tax amount in the AFC Table array.

4.3.2 Returned Tax Information

Tax information can be returned using the P Code, J Code, Zip or FIPS functions. A value is returned to indicate the number of taxes returned and included in the Tax Table. Each tax calculated is returned as a record in the tax table. Taxes are retrieved by looping through the table by the number of taxes, which is then returned by the taxation API call.

Each row in the Tax Table will contain the information shown in Table 4-12.

Table 4-12 Enhanced Taxes Table Fields	
PCode	PCode for tax jurisdiction
Tax Type	Tax Type
Tax Level	Tax Level**
Calculation Type	Calculation Type (i.e. Rate, Fixed, Per Minute, Per Line)
Rate	Tax Rate or amount applied
Tax Amount	Calculated tax amount
Taxable Measure	Amount of charge plus any taxed taxes
Exempt Sale Amount	Amount of the charge exempt from taxes
*Desc	Tax description string
Billable	Billable flag from tax logic
Compliance	Compliance flag from tax logic
Surcharge Flag	Surcharge flag from tax logic

***Country taxes within any US territory will be returned at a state level versus a federal level.*

4.3.3 Returned Tax Information Using Invoice Mode

AFC applies Steps, Brackets and/or Limits on a per transaction basis unless operating in Invoice Mode. Invoice Mode is used to group transactions that apply to the same customer. AFC maintains a history of the transactions and applies the Steps, Brackets and/or Limits to entire group of transactions.

For example, suppose a jurisdiction charges a 10% sales tax on just the first \$10 of an invoice. If two separate \$6 transactions on the invoice are passed while not in Invoice Mode, two \$0.60 tax charges would be erroneously totaled and a \$1.20 tax would be generated. However, if these same transactions are passed while in Invoice Mode, the total tax charge would be correctly totaled and a \$1.00 tax would be generated.

Table 4-13 Invoice Mode Tax Table Fields	
PCode	PCode for tax
Tax Type	Tax Type
Tax Level	Tax Level**
Calculation Type	Calculation Type (i.e. Rate, Fixed, Per Minute, Per Line)
Rate	Tax Rate or amount applied.
Tax Amount	Calculated tax amount.
Exempt Sale Amount	Amount of the charge exempt from taxes.
*Desc	Tax description string
Lines	For tax calculations based on line counts, it is the number of lines used in the calculation to produce the tax. For Federal taxes, the line count always matches the lines input. For all other taxes, it is 0.
Minutes	For tax calculations based on minutes, it is the number of minutes used in the calculation to produce the tax. For all other taxes, it is 0.
Max Base	Maximum amount to which tax is applied amounts above this will be taxed at a higher bracketed rate (if applicable).
Min Base	Minimum amount to which tax is applied amounts below this will be taxed at a lower bracketed rate (if applicable).
Excess Tax	Rate for amount above Max Base returned as part of county tax.
Total Charge	Sum of charges calculated on a per customer basis.

**Country taxes within any US territory will be returned at a state level versus a federal level.

4.3.3.1 Invoice Mode Tax Amount Adjustments

Since AFC processes each transaction as it is received with no knowledge of future transactions it must make adjustments to the tax amount when additional transactions are received that cause the rate of previous transactions to change.

Example

(Please reference and see the Rate Table for an overview of rates.)

Standard tax mode:

3 lines were received at the rate of 0.45 per line for E911 => tax amount \$1.35.

2 lines were received at the rate of 0.50 per line for E911 => tax amount \$1.00.

Total tax for E911 => \$2.35

Invoice Mode:

3 lines were received at the rate of 0.45 per line for E911 => tax amount \$1.35.

2 more lines were received for a total of 5 lines which changes the rate to 0.40 per line.

The total tax should be \$2.00 but since the customer had already been taxed \$1.35 for this same tax then the additional tax amount returned is \$0.65.

Total tax for E911 => \$2.00

In some cases the adjustment required for a rate change may cause a negative tax amount to be returned. The summarized tax amounts returned when the list of customer transactions are complete will match what the customer would have been charged had the series of transactions processed been entered as a single transaction.

4.3.4 Tax Grouping

AFC allows a client application to group the tax calculation results returned in the tax table based on tax level and tax type criteria. This may be accomplished by setting the appropriate group mode option in the configuration file (see **TM_00548_AFC Configuration Guide**). The option specified in the configuration file will automatically be applied to every AFC session as it is initialized. However, this option may be overridden by calling the EZTaxGroupResults API function.

NOTE:

This option will not modify the way that tax calculation results are logged into the EZtax log file. Only the tax calculation results returned by either one of the tax calculation API will be group according to the values set for this option.

General Rules

The following rules apply when using any value for this option:

1. Federal taxes may not be grouped. Each Federal tax will be returned individually.
2. Non-billable taxes may not be grouped.
3. Only rate-based taxes may be grouped. Taxes with a different calculation type (for example, fixed, per line, etc) will be returned individually.
4. Use taxes may be grouped with other use taxes only (for example, state and local use tax). Use taxes will not be grouped with other tax types.
5. When grouping taxes for different tax levels (for example, state and local taxes) the jurisdiction code for the lowest level jurisdiction will be returned.

6. Unincorporated taxes will be considered as County taxes when grouping taxes by tax level, and will be grouped accordingly.
7. The tax rates for all taxes being grouped into a single record will be added together.

Options:

AFC provides the following options in order to group the taxes in the tax table by tax level:

1. *Group taxes at the same level.* When using this option, taxes at the same level will be grouped together.
2. *Group state, group county and local.* By using this option, all state level taxes will be grouped into a single record, and all county and local taxes will be grouped together into a separate record.
3. *Group state, county, and local.* If this option is specified, all state, county, and local taxes will be grouped together.

In addition to grouping taxes by tax level, AFC allows you to separate sales taxes from other tax types. The following options may be used in conjunction to the tax level options specified above.

Group sales taxes. This option indicates that Sales Taxes (tax type 1) and Use Taxes should be grouped separately from other taxes.

Group taxes in the sales category. All taxes that are in the sales category (such as some District and Transit taxes, in addition to Sales Taxes) will be grouped together separately from other taxes.

If the group mode is being specified within the EZTax.cfg file, the sales tax option may be combined with the tax level option by inserting the option in the line following the tax level option. For example:

```
groupstatecountyandlocal
```

```
groupsalescategory
```

When calling the API function, a sales tax option may be combined with a tax level option by using the bitwise OR operator in the group mode parameter. For example:

```
EZTaxGroupResults(session, GROUP_ST_CO_LOCAL | GROUP_SALES_CATEGORY);
```

By using the default option in the configuration file or the API function, this feature will be disabled, and all taxes will be returned in an individual record.

Tax Return Table

When grouping taxes together, the fields in the tax return table will contain the following values:

4. Jurisdiction Code. Jurisdiction Code (PCode or JCode depending on the API call and interface being used) for the lowest level jurisdiction. For example, if Kansas state taxes and Overland Park local taxes were grouped together, the tax record will contain the jurisdiction code for Overland Park.
5. Tax level. When grouping State, County and Local taxes together, the tax record will contain a value of 6 in the tax level. When grouping only County and Local taxes together, the tax record will contain a value of 7 in the tax record. Constants are provided for these values in the appropriate file.
6. Tax type. When grouping different taxes together, the tax type in the tax record will contain a value of 0. If only Sales Taxes (tax type 1) or Use Taxes (tax type 49) are being grouped together, the tax record will contain the corresponding tax type.
7. Tax amount. This field will contain the sum of the tax amount for all taxes being grouped together.
8. Tax rate. This field will contain the sum of the tax rates for all taxes being grouped together.

NOTE:

The remaining fields in the tax table will not contain any meaningful value. Grouping tax calculation results may serve as a way to simplify the tax information for display purposes only. If further detail is required for each tax being returned by AFC, this feature should not be used.

4.3.5 Tax Types

When a transaction is processed through the AFC Engine, a Tax Type is assigned based on the results. The Tax Type is determined by the transaction location and transaction/service pairs provided in the transaction record and is used to accurately describe the exact nature of the tax applied (refer to Table 4-14).

With so many Tax Types available, highly detailed and specific tax descriptions are provided for use in taxation disclosure. This is useful in billing and customer service, and simplifies tax compliance filing.

Users can only control the tax types with the Avalara EZdata® product (sold separately). AFC allows for a change, referred to as an override (see section 4.1.3), of the rate for a specific tax type and tax level in a specific jurisdiction. When overriding a tax type, the tax type must exist in the jurisdiction where the override is to be made.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
1	Sales Tax	This is a tax on the privilege of purchasing goods and services.
2	Business and Occupation Tax	This is a tax that is normally based upon having a business, occupation, or residence within the taxing authority's geopolitical boundaries.
3	Carrier Gross Receipts	This is a tax based upon gross receipts of the telecommunications carrier. Each portion of the tax is passed on to consumers based upon the amount of their phone bill.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
4	District Tax	District taxes are taxes associated with a particular district. This is typically this is a school district; however it could be a redevelopment, sports entertainment or some other type of district.
5	Excise Tax	Excise taxes are imposed at the manufacturer and/or retail level and are virtually indistinguishable from a sales tax to the consumer. However, many excise taxes are considered part of the sale or gross receipts and are therefore taxes by sales and/or gross receipts taxes.
6	Federal Excise Tax	Federal tax on telecommunications services.
7	Fed USF A – School	A federal universal service fund imposed by the Federal Communications Commission to fund schools, libraries and rural health care support mechanisms. (See also type 18)
8	License Tax	Tax based upon the granting of a license to perform a service to the community. In many cases, this tax can be passed on to consumers.
9	P.U.C. Fee	Public Utility Commission fees are used to fund the Public Utility Commission or Public Service Commission.
10	E911 Tax	This tax is used to fund the emergency 911 systems.
11	Service Tax	This tax is used to fund a service such as the telecommunications relay service for the deaf.
12	Special Tax	Used to specify a tax that does not fit into a typical category.
13	State Universal Service Fund	The purpose of the fund depends upon the state; however these are typically used for items such as funding schools or subsidizing the cost of telecommunications users in remote locations.
14	Statutory Gross Receipts	Tax based upon the gross receipts of one or more transaction and service type combinations.
15	Surcharge	Surcharge imposed by a taxing jurisdiction upon telecommunications services. Typically these are taxed by the federal excise tax and may be taxes by other taxes as well.
16	Utility Users Tax	This is a tax imposed upon users of utilities. In this case the utility is telecommunications.
17	Sales Web Hosting	Similar to tax type 1 (Sales Tax) but applies only to web hosting services.
18	Fed Universal Service Fund	Federal Universal Service Fund charge imposed by FCC to fund schools, libraries, rural health care support mechanisms, lifeline, link-up, and the high cost fund. This charge is applied upon interstate and international telephone revenue and FCC Subscriber Line Fee charges.
19	State High Cost Fund	State high cost funds are used to subsidize the cost of telecommunications users in remote locations. For other than wireless or VoIP revenue.
20	State Deaf and Disabled Fund	This fund is used to provide access to telecommunications services for deaf and disabled individuals.
21	CA Teleconnect Fund	This fund supports California providers that offer discounts to schools, libraries, health care and community-based organizations with telecommunications services that qualify. For other than wireless or VoIP revenue.
22	Universal Lifeline Telephone Service Charge	This California state charge funds a program that provides basic telephone service to qualifying low income families. For other than wireless or VoIP revenue.
23	Telecom Relay Surcharge	Surcharge to provide funding for telecommunications access for the hearing impaired.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
24	Telecommunications Infrastructure Maintenance Fee	Fee used to fund the maintenance of telecommunications infrastructure (network, switches, etc.).
25	State Poison Control Fund	Fund used to establish and support a statewide poison control center network.
26	Telecommunications Infrastructure Fund	Fund used to support the telecommunications infrastructure.
27	NY MCTD 186c	New York Metropolitan Commuter Transportation District (NY MCTD) imposed surcharge based on the taxable telephone services subject to the NY 186e excise tax on telecommunications and said services have occurred solely within the specific NY counties that comprise the NY MCTD.
28	NY MCTD 184a	New York Metropolitan Commuter Transportation District (NY MCTD) imposed surcharge based on the taxable local telephone services subject to the NY Franchise 184 tax (tax type 45 below) and said services have occurred solely within the specific NY counties that comprise the NY MCTD.
29	Franchise Tax	Tax imposed upon a telecommunications carrier for granting of a telecommunications franchise by the governing body. Many of these can and are passed on to the consumer.
30	Utility Users Tax – Business	Similar to tax type 16 (Utility Users Tax) but applies only to business subscribers. This will occur when different rates exist for utility users based upon their being a business or residential user.
31	Fed Telecommunications Relay Service	The tax funds the Federal Telecommunications Relay Services (TRS), which is required by Title IV of the Americans with Disabilities Act.
32	District Tax (Residential)	Similar to tax type 4 (District Tax) but applies only to a residential customer.
33	Transit Tax	A specific state, county, district or local tax used to support a transportation district or program. If this is a district program, the boundaries can overlap county and local jurisdictions.
34	Telecommunications Assistance Service Fund	Similar to tax type 13 (State Universal Service Fund) but the funding is usually focused on helping low income and elderly telecommunication users. Typically this charge is accessed as a per line charge on local lines.
35	E911 Tax (Business)	Similar to tax type 10 (E911 Tax) but applies only to a business customer.
36	TRS (Business)	Similar to tax type 23 (Telecommunications Relay Service Surcharge) but applies only to a business customer.
37	Universal Service Fund (Access/Trunk line)	Similar to tax type 13 (State Universal Service Fund) but applies only on applicable local access or local trunk line.
38	Universal Service Fund (Business Line)	Similar to tax type 13 (State Universal Service Fund) but applies only on a business local line.
39	E911 Tax (PBX/Trunk line)	Similar to tax type 10 (E911 Tax) but applies only on a local PBX or local trunk line.
40	License Tax (Business)	Similar to tax type 8 (License Tax) but applies only to a business customer type.
41	Optional Telecommunications Infrastructure Maintenance Fee	Similar to tax type 24 (Telecommunications Infrastructure Maintenance Fee) but applies only in the state of Illinois and at the option of the carrier for municipalities in Illinois who do not impose a local TIMF charge.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
42	Sales Tax (Business)	Similar to tax type 1 (Sales Tax) but applies only to a business customer.
43	E911 Tax (Residential)	Similar to tax type 10 (E911 Tax) but applies only to a residential customer.
44	E911 Tax (Wireless)	Similar to tax type 10 (E911 Tax) but applies only on wireless telecommunications.
45	NY Franchise 184	NY State imposed franchise tax on local telephone carriers who are principally engaged in the conduct of local telephone business (i.e. 50% or more of the carrier's operating revenues are derived from local telephone business revenues.)
46	NY Franchise 184 Usage	Similar to tax type 45 (NY Franchise 184) but only applicable to separately charged intraLATA toll services. Said charge is still subject to the "principally engaged in the conduct of local telephone business" standard as described above in tax type 45 description.
47	NY MCTD 184a Usage	Similar to tax type 28 (NY MCTD 184a) but only applicable to separately charged intraLATA toll services. Said charge is still subject to the "principally engaged in the conduct of local telephone business" standard as described above in tax type 45 description.
48	Universal Service Fund (Wireless)	Similar to tax type 13 (State Universal Service Fund), but applies only on wireless telecommunications.
49	Use Tax	An ad Val Orem tax on the use, consumption, or storage of tangible property and usually assessed at the same rate as the sales tax of the applicable jurisdiction.
50	Sales Tax (Data)	Similar to tax type 1 (Sales Tax) but applies only on data services.
51	Municipal Right of Way	Tax imposed on local exchange telephone services to cover the municipal cost in managing and maintaining municipal rights-of-way. Typically these charges are accessed through a per line fee.
52	Municipal Right of Way (Business)	Similar to tax type 51 (Municipal Right of Way) but applies only to business customers of local exchange services.
53	Municipal Right of Way (Private Line)	Similar to tax type 51 (Municipal Right of Way) but applies only to private line customers (Residential and Business).
54	Utility Users Tax (Wireless)	Similar to tax type 16 (Utility Users Tax) but applies only on wireless telecommunications.
55	Fed USF Cellular	Similar to tax type 18 (Fed USF Combined High Cost and School) but applies only on wireless telecommunications. Following the "safe harbor" taxing and remittance standards for cellular providers set by the FCC; this tax type is assessed at a rate of thirty seven and one tenth percent (37.1%) of the current Federal USF rate as established by the FCC.
56	Fed USF Paging	Similar to tax type 18 (Fed USF Combined High Cost and School) but applies only on paging services. Following the "safe harbor" taxing and remittance standards for paging service providers set by the FCC; this tax type is assessed at a rate of twelve percent (12%) of the current Federal USF rate as established by the FCC.
57	Sales Tax (Interstate)	Similar to tax type 1 (Sales Tax) but applies only on interstate telecom services.
58	Utility Users Tax PBX Trunk	Similar to tax type 16 (Utility Users Tax) but applies only on PBX trunks.
59	District Tax Web Hosting	Similar to tax type 4 (District Tax) but applies only on web hosting services.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
60	CA High Cost Fund A	CA state fund that provides subsidies to specific small independent telephone companies. Said fund is used to minimize any rate disparity of basic telephone service between rural and metropolitan areas. For other than wireless or VoIP revenue.
61	Telecommunications Education Access Fund	State fund used to facilitate internet access and related telecom services to qualified schools and libraries.
62	Fed TRS Cellular	Similar to tax type 31 (Fed Telecommunications Relay Service) but applies only on wireless services. Following "safe harbor" taxing and remittance standards for cellular providers set by the FCC; this tax type is assessed at a rate of fifteen percent (15%) of the current Federal TRS rate as established by the FCC.
63	Fed TRS Paging	Similar to tax type 31 (Fed Telecommunications Relay Service) but applies only on paging services. Following "safe harbor" taxing and remittance standards for paging service providers set by the FCC; this tax type is assessed at a rate of twelve percent (12%) of the current Federal TRS rate as established by the FCC.
64	Communications Services Tax	A tax on end users who consume communication services.
65	Value Added Tax (VAT)	International based tax on the final consumption of certain goods and services.
66	Goods and Services Tax (GST)	Goods and service tax based on consumption.
67	Harmonized Sales Tax (HST)	Provincial sales tax applied in specific Canadian provinces. Rate is a combination of the provincial sales tax and the national GST.
68	Provincial Sales Tax (PST)	Sales tax applied in various Canadian provinces.
69	Quebec Sales Tax (QST)	Specific sales tax applied only in the province of Quebec, Canada.
70	National Contribution Regime (NCR)	National Canadian tax on telecom for the provisioning of universal service throughout Canada. Similar to the Federal USF.
71	Utility Users Tax (Cable Television)	Similar to tax type 16 (Utility Users Tax) but applies only on cable television.
72	FCC Regulatory Fee (Cable Television)	A fee used to fund the Federal Communications Commission.
73	Franchise Tax (Cable)	Similar to tax type 29 (Franchise Tax) but applies on a cable television carrier for the granting of a cable television franchise by the governing body.
74	Universal Service Fund (Paging)	Tax similar to tax type 13 (State Universal Service Fund), but applies only on paging telecommunications.
75	Statutory Gross Receipts (Wireless)	Tax similar to tax type 14 (Statutory Gross Receipts) based upon the gross receipts of one or more cellular-only transaction and service type combinations.
82	Franchise Tax (Wireless)	Similar to tax type 29 (Franchise Tax) but applies on a wireless carrier for the granting of a franchise by the governing body.
83	Reserved	Reserved
84	Public Education and Government (PEG) Access Fee	Fee to subscribers for support of PEG access.
85	Communications Service Tax (Satellite)	Similar to tax type 64 (Communications Service Tax) but applied only on satellite services.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
86	Franchise Tax (Satellite)	Similar to tax type 29 (Franchise Tax) but applies on a satellite television carrier for granting of a satellite TV franchise by the governing body.
87	Reserved	Reserved
88	Reserved	Reserved
89	TRS (Centrex)	Similar to tax type 23 (Telecommunications Relay Service Surcharge) but only applied to a Centrex extension.
90	Utility Users Tax (Cable Television – Business)	Similar to tax type 71 (Utility users Tax – Cable Television) but only applies to cable television services for business customers.
91	Utility Users Tax (Centrex)	Similar to tax type 16 (Utility users Tax) but only applies to a Centrex extension.
92	E911 (Centrex)	Similar to tax type 10 (E911) but only applies to a Centrex extension.
93	Utility Users Tax (Line)	This is a tax imposed upon users of utilities based on the number of lines. This tax should not be confused with tax type 16 (Utility Users Tax), which is based on a rate.
94	Crime Control District Tax	A specific district tax that supports a crime control program. This district can overlap county and local jurisdictions.
95	Library District Tax	A specific district tax that supports a library program. This district can overlap county and local jurisdictions.
96	Hospital District Tax	A specific district tax that supports hospital program. This district can overlap county and local jurisdictions.
97	Health Services District Tax	A specific district tax that supports a health services program. This district can overlap county and local jurisdictions.
98	Emergency Services District Tax	A specific district tax that supports an emergency services program. This district can overlap county and local jurisdictions.
99	Improvement District Tax	A specific district tax that supports a public improvement program. This district can overlap county and local jurisdictions.
100	Development District Tax	A specific district tax that supports a development program. This district can overlap county and local jurisdictions.
101	Transit Web Hosting Tax	A specific district tax on web hosting services that supports a transportation program. This district can overlap county and local jurisdictions.
102	Ambulance District Tax	A specific district tax that supports an ambulance program. This district can overlap county and local jurisdictions.
103	Fire District Tax	A specific district tax that supports a fire district. This district can overlap county and local jurisdictions.
104	Police District Tax	A specific district tax that supports a police district. This district can overlap county and local jurisdictions.
105	Football District Tax	A specific district tax that supports a football program. This district can overlap county and local jurisdictions.
106	Baseball District Tax	A specific district tax that supports a baseball program. This district can overlap county and local jurisdictions.
107	Crime Control District Web Hosting Tax	A specific district tax on web hosting services that supports a crime control program. This district can overlap county and local jurisdictions.
108	Library District Web Hosting Tax	A specific district tax on web hosting services that supports a library program. This district can overlap county and local jurisdictions.
109	Hospital District Web Hosting Tax	A specific district tax on web hosting services that supports hospital program. This district can overlap county and local jurisdictions.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
110	Health Services District Web Hosting Tax	A specific district tax on web hosting services that supports a health services program. This district can overlap county and local jurisdictions.
111	Emergency Services District Web Hosting Tax	A specific district tax on web hosting services that supports an emergency services program. This district can overlap county and local jurisdictions.
112	Improvement District Web Hosting Tax	A specific district tax on web hosting services that supports a public improvement program. This district can overlap county and local jurisdictions.
113	Development District Web Hosting Tax	A specific district tax on web hosting services that supports a development program. This district can overlap county and local jurisdictions.
114	Utility Users Tax (Interstate)	This tax is similar to tax type 16 (Utility Users Tax) but applies only to the interstate portion of transactions.
115	Utility Users Tax (Telegraph)	This tax is similar to tax type 16 (Utility Users Tax) but applies only to telegraph transactions.
116	E911 Network and Database Surcharge	Charge assessed on each access line to pay the cost of developing and maintaining a network and database for a 911 emergency system.
117	License Tax Emergency	Utility tax for emergency budgetary purposes.
118	License Tax Emergency (Business)	Utility tax for emergency budgetary purposes. (Applies to business accounts.)
119	Educational Sales Tax	Sales tax designated specifically for education and reported apart from the general sales tax.
120	Educational Use Tax	Use tax designated specifically for education and reported apart from the general use tax.
121	E911 Operational Surcharge County Commission	Portion of E911 voted upon and approved by the County Commission.
122	E911 Operational Surcharge Voter Approved	Portion of E911 voted upon and approved by the Voters in a county.
123	Sales Tax Nine Hundred	Similar to Sales Tax (Type 1) but applies to 900 calls.
124	Convention Center Tax	Sales Tax designated for convention or conference centers.
125	E911 High Capacity Trunk	This tax is used to fund the emergency 911 systems. This tax type is used in jurisdictions that have a different rate for High Capacity Trunks.
126	School Board Tax A	Tax to fund School Board. This is typically a Sales Tax. The letter designation is used in compliance reporting.
127	School Board Tax B	Tax to fund School Board. This is typically a Sales Tax. The letter designation is used in compliance reporting.
128	School Board Tax C	Tax to fund School Board. This is typically a Sales Tax. The letter designation is used in compliance reporting.
129	School Board Tax D	Tax to fund School Board. This is typically a Sales Tax. The letter designation is used in compliance reporting.
130	School Board Tax E	Tax to fund School Board. This is typically a Sales Tax. The letter designation is used in compliance reporting.
131	School Board Tax F	Tax to fund School Board. This is typically a Sales Tax. The letter designation is used in compliance reporting.
132	School District Tax	Tax to fund a School District. This is typically a Sales Tax.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
133	Police Jury Tax B	Tax to fund Police jurisdictions. This is typically a Sales Tax. The letter designation is used in compliance reporting.
134	Police Jury Tax C	Tax to fund Police jurisdictions. This is typically a Sales Tax. The letter designation is used in compliance reporting.
135	Police Jury Tax E	Tax to fund Police jurisdictions. This is typically a Sales Tax. The letter designation is used in compliance reporting.
136	Communications Services Tax (Wireless)	A tax on end users who consume communication services. This tax type applies to wireless only.
137	Service Provider Tax	Tax imposed upon the seller for providing services in a particular jurisdiction.
138	Telecommunications Sales Tax	Tax on privilege of purchasing telecommunication services. Occasionally tax jurisdictions impose an alternate sales tax rate on sales of telecommunication services. This tax type will be returned in instances where such distinction exists.
139	Advanced Transit Tax	A specific state, county, district or local tax used to support a transportation district or program. If this is a district program, the boundaries can overlap county and local jurisdictions.
140	Advanced Transit Web Hosting Tax	A specific district tax on web hosting services that supports a transportation program. This district can overlap county and local jurisdictions.
141	Missouri Universal Service Fund	Fund established by the Missouri PSC to help low-income and disabled Missourians receive discounts for basic local telephone service.
142	Businesses and Occupation Tax (Wholesale)	This is a tax that is normally based upon having a business, occupation, or residence within the taxing authority's geopolitical boundaries. This tax type pertains to wholesale transactions only.
143	Telecommunications Education Access Fund (Centrex)	State fund used to facilitate Internet access and related telecom services to qualified schools and libraries. This tax type pertains to Centrex rates only.
144	Businesses and Occupation Tax (Other)	This is a tax that is normally based upon having a business, occupation, or residence within the taxing authority's geopolitical boundaries.
145	Tribal Sales Tax	Sales tax imposed by an Indian Tribe.
146	Sales Tax (Data Processing)	This is a tax imposed on the sale of data processing services.
147	Transit Tax (Data Processing)	A specific district tax on data processing services that supports a transportation program. This district can overlap county and local jurisdictions.
148	Crime Control District Tax (Data Processing)	A specific district tax on data processing services that supports a crime control program. This district can overlap county and local jurisdictions.
149	Library District Tax (Data Processing)	A specific district tax on data processing services that supports a library program. This district can overlap county and local jurisdictions.
150	Hospital District Tax (Data Processing)	A specific district tax on data processing services that supports hospital program. This district can overlap county and local jurisdictions.
151	Health Services District Tax (Data Processing)	A specific district tax on data processing services that supports a health services program. This district can overlap county and local jurisdictions.
152	Emergency Services District Tax (Data Processing)	A specific district tax on data processing services that supports an emergency services program. This district can overlap county and local jurisdictions.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
153	Improvement District Tax (Data Processing)	A specific district tax on data processing services that supports a public improvement program. This district can overlap county and local jurisdictions.
154	Development District Tax (Data Processing)	A specific district tax on data processing services that supports a development program. This district can overlap county and local jurisdictions.
155	Advanced Transit Tax (Data Processing)	A specific district tax on data processing services that supports a transportation program. This district can overlap county and local jurisdictions.
156	CA PSPE Surcharge	Surcharge to fund the payphone tariff enforcement program.
157	District Tax (Data Processing)	District taxes are taxes associated with a particular district. Typically this is a school district; however it could be a redevelopment, sports, entertainment or some other type of district.
158	Reserved	Reserved
159	Reserved	Reserved
160	Statutory Gross Receipts (Business)	Tax based upon the gross receipts of one or more transaction and service type combinations. This tax type is returned when there is a difference between the business rate and other rates.
161	E911 Tax (VoIP)	This tax is used to fund emergency 911 systems. This tax type applies in jurisdictions that have enacted E911 charges specifically for VoIP service.
162	FUSF (VoIP)	Similar to tax type 18 (Fed USF Combined High Cost and School) but applies only on interconnected VoIP services. Following the "safe harbor" taxing and remittance standards for interconnected VoIP providers set by the FCC; this tax type is assessed at a rate of sixty four and nine tenths percent (64.9%) of the current Federal USF rate as established by the FCC.
163	FUSF	This tax type gives interconnected VoIP carriers the ability to report actual interstate/international revenues subject to the FUSF as opposed to using the Safe Harbor percentage. Those who wish to use this method of reporting should override Tax Type 162 to 0% and override Tax Type 163 to the current FUSF rate.
164	Reserved	Reserved
165	Universal Service Fund (VoIP)	Similar to tax type 13 (State Universal Service Fund), but applies only to interconnected VoIP services.
166	Communications Service Tax (Cable)	A tax on end users who consume communication services. This tax type applies to cable only.
167	Municipal Right of Way (Cable)	Tax imposed on cable television services to cover the municipal cost in managing and maintaining municipal rights-of-way. This tax type is specific to charges imposed upon residential customers of cable television services.
168	Network Access Fee – Interstate	Reserved
169	FCC Regulatory Fee (Wireline)	A fee paid by Interstate Telecommunications Service Providers to fund the Federal Communications Commission. The current rate in the system is the last rate published by the FCC and is based upon 499 revenues from a prior period; therefore it is only an estimate of actual liability. If the user does not wish to pass this fee on, or collects the fee by another mechanism, the rate can be overridden to 0%, or the user can override the rate to match their estimation.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
170	FCC Regulatory Fee (Wireless)	A fee paid by Commercial Wireless providers to fund the Federal Communications Commission. The current rate in the system is the last rate published by the FCC and is based upon the number of subscribers from a prior period; therefore it is only an estimate of actual liability. If the user does not wish to pass this fee on, or collects the fee by another mechanism, the rate can be overridden to 0%, or the user can override the rate to match their estimation.
171	Reserved	Reserved
172	Statutory Gross Receipts (Video)	Tax based upon the gross receipts of video services such as cable or satellite.
173	Utility Users Tax – Lifeline	Similar to tax type 16 (Utility Users Tax) but only applies to lifeline customers.
174	TRS – Long Distance	Similar to tax type 23 (Telecommunications Relay Service) but applies exclusively to long distance.
175	Telecom Relay Surcharge (Wireless)	Similar to Tax Type 23 (Telecom Relay Surcharge), but applied only to wireless.
176	Sales Tax – Senior Citizen	Similar to tax type 1 (Sales Tax) but only applies to Senior Citizens who meet certain age requirements.
177	Regulatory Cost Charge – Local	Fee charged by the applicable regulatory agency to cover that agencies expenses for the upcoming year. This fee covers local phone service only.
178	Regulatory Cost Charge – Intrastate	Same as Regulatory Cost Charge-Local. The fee would cover intrastate calling only.
179	Regulatory Cost Charge – Cable	Same as Regulatory Cost Charge-Local. The fee would cover cable services only.
180	P.U.C. Fee – Cable	Similar to Tax Type 9, but applies to Cable television revenues only.
181	Provincial Sales Tax	Sales Tax applied in various Canadian Provinces. This Tax Type applies to TOLL services only.
182	Reserved	Reserved
183	Reserved	Reserved
184	Sales Tax-Manufacturing	Refers to a sales tax rate charged on the sale of manufacturing machinery and other manufacturing related items.
185	Use Tax-Manufacturing	Refers to a use tax rate charged on the sale of manufacturing machinery and other manufacturing related items.
186	Sales Tax-Motor Vehicles	Refers to a sales tax rate charged on the sale of motor vehicles.
187	Use Tax-Motor Vehicles	Refers to a use tax rate charged on the sale of motor vehicles.
188	Rental Tax	Tax exclusively on the rental of any item not specifically taxed by another rental tax.
189	Rental Tax-Linen	Tax covering the rental of linen based supplies.
190	Sales Tax-Vending	Sales tax that applies to the retail sale of items sold through vending machines.
191	Rental Tax-Motor Vehicles	Tax covering the rental of motor vehicles.
192	Sales Tax-Wholesale	Sales Tax applying to wholesale transactions.
193	Sales Tax-Food and Drugs	Refers to a rate charged on the sale of food, drugs or beverages.
194	Sales Tax-Food	Refers to a rate charged on the sale of food or beverages.
195	Fur Tax	Tax charged on the sale of furs.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
196	Privilege Tax-Manufacturing	Tax on the privilege of purchasing items to be used in the manufacturing process.
197	Lead Acid Battery Fee	Fee charged to cover the cost involved in the disposing of lead based batteries.
198	Sales Tax-Motor Fuel	Refers to a sales tax rate charged on the sale of motor fuel.
199	Lead Acid Battery Fee-Larger Battery	Fee charged for batteries over a certain pre-described voltage to cover the cost involved in disposing lead based batteries.
200	Sales Tax-Parking	Tax on the fee charged for the parking of motor vehicles.
201	Privilege Tax-Recreation	Tax charged for recreational events such as sporting events or any similar type of endeavor.
202	Dry Cleaning Fee	Fee charged on the sale of dry cleaning services.
203	White Goods Tax	A fee applied to the sale of certain appliance and appliance type items to cover the disposal of such items.
204	Sales Tax-Medical Equipment	Sales Tax that applies exclusively to the sale of medical equipment.
205	Electronic Waste Recycling Fee-Small	A fee charged for smaller monitors to cover the disposal of such items.
206	Electronic Waste Recycling Fee-Medium	A fee charged for certain sized monitors fitting between certain dimensions to cover the disposal of such items.
207	Electronic Waste Recycling Fee-Large	A fee charged for larger monitors to cover the disposal of such items.
208	Alcoholic Beverage Tax	Alcoholic Beverages taxed under a different tax in lieu or in addition to sales tax.
209	Sales Tax-Alcohol	Refers to a sales tax rate charged on the sale of alcohol.
210	Liquor Drink Tax	Applies where there is a distinct rate on the sale of mixed drinks ready for on-site consumption.
211	IN Universal Service Charge.	Indiana fee used to fund schools and underserved areas of the state.
212	TRS (Paging)	Similar to Tax Type 23 (Telecommunications Relay Service Surcharge) but only applied to paging.
213	ConnectME Fund	Assessment on state revenues to provide service to underserved areas of Maine.
214	PA PURTA Surcharge	Similar to Tax Type 14 in PA, but this surcharge applies only to intrastate revenues.
215	ConnectME Fund(VoIP)	Similar to Tax Type 213 in ME, but this surcharge applies only to VoIP.
216	ConnectME Fund(Cable)	Similar to Tax Type 213 in ME, but this surcharge applies only to Cable
217	TRS VoIP	Similar to Tax Type 23 (Telecommunications Relay Service Surcharge), but applies only to VoIP
218	Consumer Council Fee	This fee provides funding for the Consumer Counsel, which represents public utility consumers before the PSC and similar groups in matters concerning public utility regulation.
219	San Diego Underground Conversion Surcharge	Surcharge imposed to pay for costs associated with undergrounding aerial telephone facilities in San Diego pursuant to the City of San Diego Underground Utilities Procedural Ordinance and the San Diego Surcharge for Underground Conversion Costs.
220	RSPF Surcharge	An Oregon Surcharge that funds the Telecommunications Relay Service, Telephone Assistance Program, and the Telecommunications Devices Access Program.
221	Reserved	Reserved

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
222	Reserved	Reserved
223	CASF	California Advanced Services Fund was created to encourage deployment of broadband facilities for use in provisioning advanced telecommunications service in unserved or underserved areas of California. For other than wireless or VoIP revenue.
224	License Tax (Cable)	Similar to Tax Type 8 (License Tax), but applies to providers of cable television services.
225	Relay Missouri Surcharge	A Missouri surcharge that funds the State Deaf and Disabled Fund.
226	FCC Regulatory Fee (VoIP)	Similar to Tax Type 169 (FCC Regulatory Fee (Wireline)), but applies to VoIP services.
228	Municipal Right of Way (Extension)	Similar to Tax Type 51 – Right of Way (Residential), but only applies to Centrex and PBX extensions.
227	Reserved	
229	Carrier Cost Recovery (VoIP)	
230	Sales Tax-Video	A sales tax charged on the provision of video services.
231	North Carolina Telecommunications Sales Tax	Tax on privilege of purchasing telecommunication services in North Carolina.
232	Telecommunications Relay Surcharge (Cellular)	Similar to tax type 31 (Fed Telecommunications Relay Service) but applies only on wireless services. This tax type is assessed at the "safe harbor" taxing and remittance standards for cellular providers set by the FCC.
233	E-911 Prepaid Wireless	Similar to Tax Type 10 (E911 Tax) but applies only on prepaid wireless telecommunications.
234	Telecommunications Relay Surcharge (Paging)	Similar to tax type 31 (Fed Telecommunications Relay Service) but applies only on paging services. This tax type is assessed at the "safe harbor" taxing and remittance standards for paging providers set by the FCC.
235	Telecommunications Relay Surcharge (VoIP)	Similar to tax type 31 (Fed Telecommunications Relay Service) but applies only on VoIP services. This tax type is assessed at the "safe harbor" taxing and remittance standards for VoIP providers set by the FCC.
236	TDAP	Program designed to distribute appropriate telecommunications devices so that persons who have a disability may effectively use basic telephone service.
237	TAP Surcharge	Surcharge to provide funding for telecommunications device for the deaf distribution program.
238	Communications Service Tax (Non-Facilities)	Similar to Tax Type 64, but applies only to providers without facilities in the public right-of-way.
239	E-911 (VoIP) Alternate	This tax type gives interconnected VoIP carriers the ability to report actual interstate/international revenues subject to the E-911 as opposed to using the Safe Harbor percentage. Those who wish to use this method of reporting should override Tax Type 161 to 0% and override Tax Type 239 to the current E-911 rate.
240	E-911 (VoIP PBX)	Similar to Tax Type 10 (E911) but applies only to VoIP PBX Service.
241	Utility Users Tax (VoIP)	Similar to tax type 16 (Utility Users Tax) but applies only to users VoIP services.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
242	Utility Users Tax (VoIP-Business)	Similar to tax type 241 (Utility Users Tax (VoIP)) but applies only to VoIP business subscribers. This tax type will be used when different rates exist for utility users based upon business versus residential use.
243	Solid Waste Collection Tax	Tax on the service of removing solid waste.
244	E-911 (VoIP Business)	This tax is used to fund emergency 911 systems. This tax type applies in jurisdictions that have enacted E911 charges specifically for VoIP service. It applies only to a business customer.
245	E-911 (VoIP Nomadic)	Similar to tax type 10 (E911) but applies only to a nomadic VoIP customer.
246	E-911 Prepaid Wireless (Alternate)	This tax type gives prepaid wireless providers the ability to report E-911 charged at the point of sale as opposed to using a fixed amount for every \$X of service. Those who wish to use this method of reporting should override Tax Type 233 to \$0 and override Tax Type 246 to the current E-911 rate.
247	Police and Fire Protection Fee	A tax to fund public safety services.
248	San Francisco Access Line Tax	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.
249	San Francisco Access Line Tax (PBX/Trunk Line)	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.
250	San Francisco Access Line Tax (VoIP)	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.
251	San Francisco Access Line Tax (Wireless)	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.
252	San Francisco Access Line Tax (High Cap Trunk)	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.
253	City of San Jose Telephone Line Tax	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.
254	City of San Jose Telephone Line Tax-PBX/Trunk Line	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.
255	City of San Jose Telephone Line Tax (VoIP)	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.
256	City of San Jose Telephone Line Tax (Wireless)	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.
257	San Leandro Emerg Com Sys Access Tax	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
258	San Leandro Emerg Com Sys Access Tax (PBX Trunk)	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.
259	San Leandro Emerg Com Sys Access Tax (VoIP)	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.
260	San Leandro Emerg Com Sys Access Tax (Wireless)	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.
261	San Leandro Emerg Com Sys Access Tax-High Cap Trnk	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.
262	Police and Fire Protection Fee (Prepaid)	A tax to fund public safety services. This applies only to prepaid services.
263	Public Safety Communications Surcharge	A tax to fund public safety services.
264	E-911 Technical Charge	A charge applied to users of E-911 services that is retained by the carrier to absorb costs incurred for the provision of E-911 service.
265	Telecom Assistance Svc Fund-High Capacity Trunk	Similar to Tax Type 34 but applies only to High Capacity Trunks
266	Reserved	Reserved
267	Access Line Tax	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.
268	Access Line Tax (PBX/Trunk Line)	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.
269	Access Line Tax (VoIP)	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.
270	Access Line Tax (Wireless)	A tax to fund such general fund services as may be determined by the Board of Supervisors including, without limitation, police, fire and emergency services.
271	WI USF	A fund in the state of Wisconsin to fund subsidizing the cost of telecommunications users in remote locations.
272	Network Access Fee – Interstate	Reserved
273	Sales Tax – Other	Refers to a separate sales tax rate charged on transactions that do not fall into another existing category.
274	FCC Regulatory Fee (VoIP Alternate)	Reserved
275	Excise Tax (Wireless)	Similar to Tax Type 5, but applies only to a wireless customer.
276	Reserved	Reserved
277	Federal Universal Service Fund (Non-billable)	Charge imposed by the FCC to fund schools, libraries, rural healthcare support mechanisms, lifeline, link-up and high cost fund. Charges included in this tax type are liabilities of the company, but can not be billed to the end-user.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
278	Municipal Right of Way-High Capacity Trunk	Similar to tax type 51 (Municipal Right of Way (Residential)) but applies only to High Capacity Trunks.
279	Education Cess	A tax levied to collect funds for education.
280	Secondary and Higher Education Cess	A tax levied to collect funds for secondary and higher education.
281	Utility Users Tax (Video)	Similar to Tax Type 16 (Utility Users Tax) but applies only on Video services.
282	State USF (VoIP Alternate)	This tax type gives the user the ability to report VoIP usage based on actual usage rather than using the Safe Harbor percentage.
283	TRS (VoIP Business)	Similar to Tax Type 23 (Telecommunications Relay Service Surcharge) but applies only to VoIP services for business customers.
284	TRS (Trunk)	Similar to Tax Type 23. (Telecommunications Relay Service Surcharge) but applies only to Trunk type services.
285	Deaf and Disabled Fund (Wireless)	Similar to Tax Type 20 (State Deaf and Disabled Fund), but applies only to wireless transactions.
286	UUT-Wireless (Business)	Similar to Tax Type 16, but applies only to Wireless Business subscribers.
287	Telecommunications Sales Tax-Prepaid	Special Sales Tax used for Telecommunications Services applied to prepaid services only.
288	CA High Cost Fund A (VoIP Actual)	This tax type gives the user the ability to report VoIP usage based on actual usage rather than using the Safe Harbor percentage.
289	State High Cost Fund (VoIP Actual)	This tax type gives the user the ability to report VoIP usage based on actual usage rather than using the Safe Harbor percentage.
290	Universal Lifeline Telephone Svc Chg (VoIP Actual)	This tax type gives the user the ability to report VoIP usage based on actual usage rather than using the Safe Harbor percentage.
291	Telecommunications Relay Svc Charge (VoIP Actual)	This tax type gives the user the ability to report VoIP usage based on actual usage rather than using the Safe Harbor percentage.
292	CA Teleconnect Fund (VoIP Actual)	This tax type gives the user the ability to report VoIP usage based on actual usage rather than using the Safe Harbor percentage.
293	CASF (VoIP Actual)	This tax type gives the user the ability to report VoIP usage based on actual usage rather than using the Safe Harbor percentage.
294	Oklahoma Sales Tax	Similar to Tax Type 1 (Sales Tax) but applies only in Oklahoma.
295	Business and Occupation Tax (Printing/Publishing)	Similar to Tax Type 2, but applied only to Printing and Publishing services.
296	Premier Resort Area Tax	Similar to Sales Tax (Tax Type 1) , but applied only in Premier Resort Areas.
297	911 Equalization Surcharge	A surcharge that provides funding for the emergency 911 system.
298	Universal Service Fee	The purpose of the fund depends upon the state, however these are typically used for items such as funding schools or subsidizing the cost of telecommunications users in remote locations.
299	NE Universal Service	Similar to Tax Type 13. But, it is applied only in the state of Nebraska.
300	TAP Surcharge Wireless	Similar to Tax Type 237 (TAP Surcharge) but applied only to wireless services.
301	GA Universal Access Fund	Similar to Tax Type 13, used in GA to provide funding for the cost of telecommunications in remote areas.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
302	CA High Cost Fund A (Wireless)	Similar to Tax Type 60. Used only for wireless service.
303	CA Teleconnect Fund (Wireless)	Similar to Tax Type 21. Used only for wireless service.
304	CASF (Wireless)	Similar to Tax Type 223. Used only for wireless service.
305	State High Cost Fund (Wireless)	Similar to Tax Type 19. Used only for wireless service.
306	PUC Fee (Wireless)	Similar to Tax Type 9. Used only for wireless service.
307	Universal Lifeline Telephone Svc Charge (Wireless)	Similar to Tax Type 22. Used only for wireless service.
308	NY TAF	Targeted Accessibility Fund of New York. Used to ensure proper funding of Lifeline, E 911, Public Interest Pay Phones and TRS.
309	Prepaid Wireless E911 TRS Surcharge	A surcharge for E911 and TRS which is applied to Prepaid Wireless Service.
310	TRS – Prepaid Wireless	Similar to Tax Type 23, but applied only to Prepaid Wireless services.
311	FUSF (Multi-line)	Federal Universal Service Fund charge imposed by FCC to fund schools, libraries, rural health care support mechanisms, lifeline, link-up, and the high cost fund. This charge is applied upon FCC Subscriber Line Charges applicable to multi-line local phone service. This tax type allows for regulatory compliance with FCC regulations in regards to multi-line accounts.
312	ND Gross Receipts Tax	A North Dakota tax based upon the gross receipts of one or more transaction and service type combinations.
313	NY Sales Tax	Similar to Tax Type 1 (Sales Tax) but applied only in New York.
314	NY Local Transit Tax	Similar to Tax Type 33 (Transit Tax) but applied only in New York.
315	NY Local District Tax	Similar to Tax Type 4 (District Tax) but applied only in New York.
316	Sales Tax - Satellite	Similar to Tax Type 1, but applied only to satellite service.
317	Sales Tax - Commercial Lease	Similar to Tax Type 1, but applied only to commercial leases.
319	Network Access Fee LD – Interstate	Reserved
320	Network Access Fee LD – Intrastate	Reserved
393	Tasa de Control	A telecommunications regulatory fee applied in Argentina.
394	Radio Rights Fee	A fee applied per station and per frequency to providers of Mobile Telecommunications.
397	Montana Excise Tax	Similar to Excise Tax (Tax Type 5) but applied only in Montana.
398	Rural Transportation Authority District Tax	A specific district tax that supports a Rural Transportation Authority.
399	MHA District Tax	A specific district tax that supports a Multi-jurisdictional Housing Authority.
400	Public Safety Improvements District Tax	A specific district tax that supports public safety improvements.
401	Mass Transit District Tax	A specific district tax that supports Mass Transit.
402	Metropolitan District Tax	A specific district tax that supports a Metropolitan district.
409	VAT (Reduced Rate)	Similar to VAT (Tax Type 65) but applied at a reduced rate.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
410	Poison Control Fund (Wireless)	Similar to Poison Control Fund (Tax Type 25) but applied only to Wireless transactions
411	State Inspection and Supervision	Reserved
424	Revenue Statement	Reserved
425	NY MCTD 186c (Wireless)	New York Metropolitan Commuter Transportation District (NY MCTD) imposed surcharge based on the taxable wireless telephone services subject to the NY 186e excise tax on telecommunications and said services have occurred solely within the specific NY counties that comprise the NY MCTD.
426	WY USF	The purpose of the fund depends upon the state; however, these are typically used for items such as funding schools or subsidizing the cost of telecommunications users in remote locations. This USF tax type does not tax the FUSF.
427	WY USF (Paging)	Tax similar to tax type 13 (State Universal Service Fund), but applies only on paging telecommunications. This USF tax type does not tax the FUSF.
428	WY USF (Wireless)	Similar to tax type 13 (State Universal Service Fund), but applies only on wireless telecommunications. This USF tax type does not tax the FUSF.
429	FCC Regulatory Fee-Toll Free	A fee paid by interstate telecommunications providers to fund the Federal Communications Commission. The current rate is the last rate published by the FCC and is based upon the number of subscribers from a prior period and is an estimate of actual liability.
430	FCC Regulatory Fee (Satellite)	A fee paid by Satellite Television Service providers to fund the Federal Communications Commission. The current rate in the system is the last rate published by the FCC and is based upon the number of subscribers from a prior period, therefore it is an estimate of actual liability. If the user does not wish to pass this fee on, or collects the fee by another mechanism, the rate can be overridden to 0%, or the user can override the rate to match their estimation.
431	Commerce Tax	Tax on Gross Revenue for the privilege of engaging in business.
432	Telecom Assistance Svc Fund - VoIP	Similar to Tax Type 34, but applies only to VoIP lines.
433	Telecom Assistance Svc Fund - VoIP High Cap Trnk	Similar to Tax Type 34, but applies only to VoIP High Capacity Trunks.
434	E-911 (VoIP Nomadic PBX)	Similar to Tax Type 10 (E911), but applies only to VoIP Nomadic PBX Service.
435	E-911 Service Fee (NL 911 Bureau)	Newfoundland and Labrador's Provincial E911 Fee.
436	Copyright Fee (Rated)	Fee to compensate copyright owners for re-transmission of copyrighted programs. Customer charged as a percentage of revenue.
437	Copyright Fee (Fixed)	Fee to compensate copyright owners for re-transmission of copyrighted programs. Customer charged as a fixed amount.
438	Utility Tax	This is a tax imposed upon utility services. In this case, the utility is telecommunications.
439	Audio-Video Service Tax	This is a tax imposed upon service provided by a multi-channel video or audio service provider.
440	Swachh Bharat Cess	A tax to collect funds for the Swachh Bharat (Clean India) Initiative.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
441	PIS	A social contribution tax targeted to finance unemployment insurance and allowance for low paid workers.
442	COFINS	A contribution levied to finance social security, health and social care.
443	ICMS	State tax for goods and services.
444	Federal USF (Centrex)	Federal Universal Service Fund charge imposed by FCC to fund schools, libraries, rural health care support mechanisms, lifeline, link-up, and the high cost fund. This charge is applied upon FCC Subscriber Line Charges applicable to Centrex service. This tax type allows for regulatory compliance with FCC regulations in regards to Centrex accounts.
445	UUT (Prepaid Wireless)	Similar to Utility Users Tax (UUT) (Tax Type 16) but only applied to prepaid wireless telecommunications.
446	Mobile Telephony Services Surcharge	Mobile Telephony Services (MTS) Surcharge on prepaid wireless telecommunications recovering California PUC expenses on wireless communication services.
447	Access Line Tax (Prepaid Wireless)	Similar to Access Line Tax (Tax Type 267) but only applied to prepaid wireless telecommunications.
448	San Leandro Emerg Com Sys Acc Tax (Ppd Wireless)	Similar to San Leandro Emerg Com Sys Access Tax (Tax Type 257) but only applied to prepaid wireless telecommunications.
449	Rental Tax (Lower Rate)	Similar to Rental Tax (Tax Type 188) but only applied to certain items at a reduced rate.
450	CA High Cost Fund A (VoIP)	CA state fund that provides subsidies to specific small independent telephone companies. Said fund is used to minimize any rate disparity of basic telephone service between rural and metropolitan areas. This Tax Type is for reporting VoIP revenues.
451	State High Cost Fund (VoIP)	State high cost funds are used to subsidize the cost of telecommunications users in remote locations. This Tax Type is for reporting VoIP revenues.
452	CA Teleconnect Fund (VoIP)	This fund supports California providers that offer discounts to schools, libraries, health care and community-based organizations with telecommunications services that qualify. This Tax Type is for reporting VoIP revenues.
453	CASF (VoIP)	California Advanced Services Fund was created to encourage deployment of broadband facilities for use in provisioning advanced telecommunications service in unserved or underserved areas of California. This Tax Type is for reporting VoIP revenues.
454	Universal Lifeline Telephone Service Charge (VoIP)	This California state charge funds a program that provides basic telephone service to qualifying low income families. This Tax Type is for reporting VoIP revenues.
455	FUNTTEL	A telecommunications tax levied on providers of telecommunications services to encourage the process of technological innovation.
456	FUST	Fund of Universalization of Telecommunication Services imposed to cover costs of universal telecommunications services. It is applied on the gross operating revenue resulting from the provision of telecommunication services.
457	Telecommunications Use Tax	Special use tax used for compliance reporting.
458	Krishi Kalyan Cess	A tax to generate funds for financing and promoting agricultural improvement initiatives.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
459	School and Library Fund Surcharge	Required invoice presentation for the Maine Telecommunications Education Access Fund. This is a state fund used to facilitate internet access and related telecom services to qualified schools and libraries.
460	State 911 Charge	Provides funding for the state emergency 911 system.
461	ITAC Assessment	The Illinois Telecommunications Access Corporation (ITAC) Assessment assists persons that have a hearing or speech disability.
462	State 911 Charge (Wireless)	Provides funding for the state emergency 911 system but applies only on wireless telecommunications.
463	E-911 (Advanced Services)	Similar to Tax Type 10 (E-911) but applied to Advanced Services.
464	VAT (Wireless)	Similar to Tax Type 65 (VAT) but only applied only to wireless services.
465	VAT (Communications)	Similar to Tax Type 65 (VAT) but only applied to communications services.
466	CA TRS	Similar to Tax Type 23 (TRS) but only applied in California.
467	CA TRS (Wireless)	Similar to Tax Type 175 (TRS (Wireless)) but only applied in California. This tax type is assessed at the safe harbor taxing and remittance standards for wireless providers set by the FCC.
468	CA PUC Fee	Similar to Tax Type 9 (PUC Fee) but only applied in California.
469	Use Tax (Rental)	Similar to Tax Type 49 (Use Tax) but only applied to rental services.
470	Use Tax (Other)	Similar to Tax Type 49 (Use Tax) but only applied to transactions that do not fall into another existing tax type. Generally transactions return this tax type if a distinct use tax rate applies in a specific jurisdiction or on a temporary basis.
473	SC USF	Similar to Tax Type 13 (State USF) but only applied in South Carolina.
474	USF (Prepaid Wireless)	Similar to Tax Type 13 (State USF) but only applied to prepaid wireless services.
475	E-911 (Lifeline)	Similar to Tax Type 10 (E-911) but only applied to a lifeline customer.
476	Utility Tax NF	Similar to Tax Type 438 (Utility Tax), but does not include Federal USF and Federal FCC Regulatory Fees in the assessment base.
477	Telecommunications Sales Tax (Wholesale)	Special sales tax used for sales of telecommunications services made on wholesale basis.
478	E-rate Broadband Program	E-rate Broadband Program to assist schools and public libraries to fund broadband facilities and obtain broadband connectivity.
479	E-rate Broadband Program (Business Line)	Similar to tax type 478 (E-rate Broadband Program) but applies only to a business local line.
480	E-rate Broadband Program (Line)	Similar to tax type 478 (E-rate Broadband Program) but applies only to applicable local access or local trunk line.
481	E-rate Broadband Program (Wireless)	Similar to tax type 478 (E-rate Broadband Program) but applies only to wireless telecommunications.
482	IGST (Communications)	Integrated goods and service tax based on consumption but applied only to communications services.
483	CGST	National goods and service tax based on consumption within the boundary of a state or territory.
484	CGST (Communications)	Similar to Tax Type 483 (CGST) but applied only to communications services.
485	SGST	State goods and service tax based on consumption within the boundary of a state or territory.
486	SGST (Communications)	Similar to Tax Type 485 (SGST) but applied only to communications services.

Table 4-14 Tax Types Supported by AFC

TAX TYPE ID	NAME	DESCRIPTION
487	Universal Service Fund (Other)	Similar to Tax Type 13 (State Universal Service Fund) but only applied to transactions that do not fall into another existing tax type.
488	IGST	Integrated goods and service tax based on consumption.
489	Kentucky Lifeline Surcharge	Kentucky Universal Lifeline Telephone Service surcharge to provide basic telephone service to qualifying low income families. This is the required invoice presentation.
490	Telecommunications Sales Tax NF	Special Sales Tax used for Telecommunications Services, but does not include Federal USF and Federal FCC Regulatory Fees in the assessment base.
491	Public Safety Communications Surcharge (Prepaid)	A tax to fund public safety services but only applied to prepaid wireless service.
492	Statutory Gross Receipts NF	Similar to Tax Type 14 (Statutory Gross Receipts), but does not include Federal USF and Federal FCC Regulatory Fees in the assessment base.

For the most current list of Tax Types, refer to the EZTaxTaxType.h file located on the most recent Distribution/Update. For .NET users, the tax types are also available in EZTaxConstants.cs. For Java users, the tax types are also available in TaxType.java.

4.3.6 Nexus

AFC allows a client application to set the nexus information in multiple methods.

- Set the appropriate nexus option in the configuration file (see [TM_00548_AFC Configuration Guide](#)). The option specified in the configuration file will automatically be applied to every AFC session as it is initialized. However, this option may be overridden by calling the EZTaxSetNexus API function.
- By EZTaxSetNexus API function.
- By using a nexus_table at the transaction level.
- By EZTaxSetStateNexus API function.

General Usage

When the engine determines there is no nexus for the taxing jurisdiction of the transaction, no taxes will be returned.

If the nexus information has been set by either configuration file or by API call, the transaction level nexus will take precedence.

EZTaxSetNexus allows one call to set all nexus information whereas EZTaxSetStateNexus allows the client application to set each state individually.

By using EZTaxSetNexus with a null pointer and a nexus count of zero, it will reset the AFC engine to default value of nexus in every state.

Nexus is only used within the United States.

4.3.7 Exclusions

AFC allows a client application to exclude states or countries (and all lower level jurisdictions) from taxation. By using the exclusion option, the client is informing AFC that the specified state or country and all lower jurisdictions are to return no taxes. An excluded state will exempt all state and lower level jurisdiction taxes; however, it will continue to return federal level taxes. An excluded country will exempt all federal and lower level jurisdiction taxes. Exclusions may be accomplished using several methods, such as those listed below:

- Set the appropriate exclusion option in the configuration file (see [TM_00548_AFC Configuration Guide](#)). The option specified in the configuration file will automatically be applied to every AFC session as it is initialized. However, this option may be overridden with other API functions.
- Use the **EZTaxSetStateExclusion** API function.
- Use the **EZTaxClearExclusion** API function.

General Usage

When the engine determines that taxes are excluded at the state level for a transaction, only federal level taxes are returned. When the engine determines that taxes are excluded at the federal level, no taxes are returned.

EZTaxClearExclusion will clear all exclusion information and return taxes at every level.

US Territories

US Territory exclusions function like state exclusions and are set and cleared when the federal level USA country code is used without a state or territory. When specified as a state jurisdiction using the two letter state code, ex. 'USA,VI', the exclusion behaves like any other state jurisdiction.

Country codes for US Territories are deprecated.

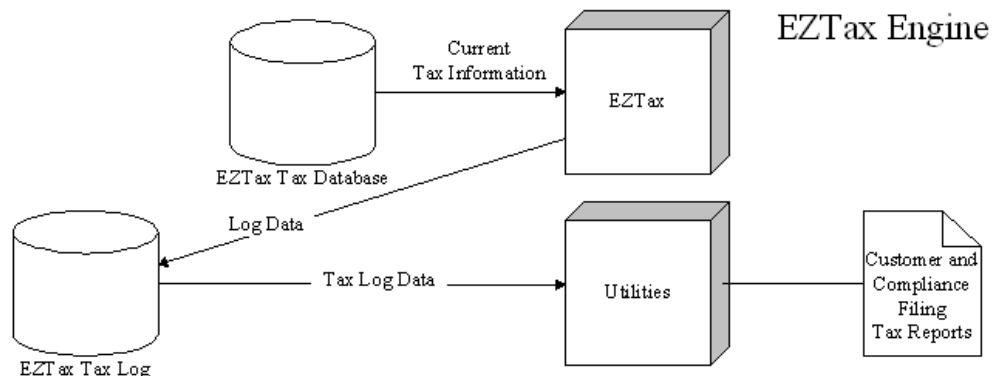
5. Utilities

Refer to Figure 5-1Figure 5-1. While tax results are typically returned to the billing system using functions, the primary output mechanism of the AFC Engine employs the EZtax Log. The EZTax.log is the link to the outside world. There are twenty-four utilities to choose from, one of which will almost certainly suit your need.

When an AFC session is started with the AFC Initialization Function, the Tax Log is opened if the Tax Log parameter is set to true. This allows the AFC Log to capture the binary formatted transaction and tax data generated by the AFC Engine.

After exiting the AFC session, the AFC Log becomes available for use. This file should be archived by the user for reruns, recovery and auditing purposes. This is accomplished by simply copying the EZTax.log file to a local storage location (refer to Figure 5-1). In most cases this binary file will be used in the future and nearly all of the utilities will delete the contents of the AFC Log upon job completion.

Figure 5-1 AFC Log Data Path



If the Billing System is configured to use the AFC Functions and does not require the logging of data, set the first parameter of the AFC Initialization Function to FALSE to disable transaction logging. For instance, if your program is saving or using the tax data returned to it by the tax calculation functions, you would have no need to use the data in the log file for later processing.

NOTE:

When running AFC in batch fashion the EZTax.log file is always generated.

It is imperative that rigid backup practices are established and followed to avoid losing the transaction and data information created during a session. Most utilities will delete the inputted log file when the utility is finished and backups would be required should reruns be needed.

5.1 Utility Selection

AFC provides utilities for reporting and file management. The latest utilities, along with their file formats, are located on the most current update. AFC has conversion utilities available for converting the tax log file to formats used by compliance services. Contact Avalara for information on converting to specific services.

5.1.1 Reporting Utilities

The reporting utilities are used for customer billing and compliance reporting. They extract specific data from the content rich binary EZtax Log file, commonly sorting the data into a logical and orderly format. The resulting output is a flat file containing just the information required for the specific report, compatible with common software such as spreadsheets and databases.

The most common use of this information is to file tax returns. However, other uses may include audit trails, internal tracking and projections.

5.1.1.1 Reporting File Extensions

Refer to Table 5-1. The reporting utility output file extensions are customized to indicate the type of information provided within the file. Note that some utilities may produce more than one type of output file.

Table 5-1 File Extension Definitions

Log File Extension	Log File Type	File Format	Note
.csf	Customer Billable amounts	.csf	
.ssf	Tax Compliance amounts	.ssf	This file is not generated when the [-cl] command line option is used.
.nba	Non-billable amounts for compliance	.ssf	This file is not generated when the [-cl] command line option is used. If there are no non-billable amounts, this file is not created.
.nca	Non-compliance amounts that were billed	.ssf	This file is generated only if there are transactions in the AFC log that do not require compliance with a jurisdiction.
.scl	Billable amounts combined with non-billable amounts	.ssf	This file is generated when the [-cl] command line option is used. The .scl file is created instead of the .ssf and .nba files.

The report file generated by the utility will be formatted as a comma delimited file or fixed field length file depending on the utility used. Refer to the specific utility documentation to determine which formatting will be provided.

Some report utilities provide specialized reports with the file name log.rpt. These comma delimited files are modifications of .csf and .ssf report formats.

Refer to Figure 5-2. For example, the EZLog_NS.exe utility produces a comma-delimited flat file report with an .ssf extension. When this file is copied into a preformatted (headings and column widths preset) MS Excel spreadsheet the following output is displayed. If the spreadsheet had not been preformatted the titles would not be present and the field widths might vary.

Figure 5-2 Sample EZLog_NS spreadsheet

<u>Country</u>	<u>State</u>	<u>County</u>	<u>Locality</u>	<u>Tax type</u>	<u>Tax level</u>	<u>Tax rate</u>	<u>Tax amount</u>	<u>Gross sales</u>	<u>Exempt</u>	<u>Adj.</u>	<u>Taxable Measure</u>	<u>Minutes</u>
USA				6	0	0.0300	29,881.87	996,062.25	0	0	996,062.25	14,859,213
USA				7	0	0.0076	10,116.13	1,331,070.00	0	0	1,331,070.00	20,410,578
USA				18	0	0.0314	11,145.49	354,952.00	0	0	354,952.00	5,446,827
USA	AL			16	1	0.0670	969.97	14,477.20	0	0	14,477.20	220,239
USA	AZ			1	1	0.0500	0.81	16.12	0	0	16.12	245
USA	AZ			10	1	0.0125	9.07	725.47	0	0	725.47	11,038
USA	AZ			12	1	0.0110	7.98	725.47	0	0	725.47	11,038
USA	AZ	APACHE		1	1	0.0500	0.81	16.12	0	0	16.12	245
USA	AZ	APACHE		1	2	0.0270	0.44	16.12	0	0	16.12	245
USA	AZ	APACHE	SPRINGERVILLE	1	1	0.0500	0.71	14.11	0	0	14.11	215

5.1.2 File Management Utilities

File management utilities are provided for maintenance purposes, such as combining one or more archived AFC log files or appending one log file to another.

Note that AFC provides the ascilog.exe utility for maintenance of AFC log files. It allows the user to view the unsorted log file as readable ASCII text without deleting the file.

5.2 Specifying a Log File at Run Time

The AFC Log file to be submitted to a utility is specified by the log file name in the Filelocs.txt file (EZTax.log by default). However, this file name will be ignored if a different file name is provided in the AFC Initialization Function at session startup.

The Filelocs.txt contains the file locations for text file contains all of paths to the AFC Data Base files. Each file in the AFC Data base is represented by one line. The lines in the text file must be in the defined order and all files must be represented.

NOTE: Batch utility clients can only use the filelocs.txt file as they will not be using the function programming method.

5.3 Log.sum File

The Log.sum file accumulates tax compliance totals over multiple runs to generate a compilation of the tax totals. It contains a summary of the EZTax.log(s) with the data organized by taxing jurisdictions. Only the EZ* prefix report utilities can be used to generate the log.sum file, which can then be passed as the input to most of the report utilities.

This is a turnaround file used for maintaining one smaller log file rather than multiple log files in cases where the customer billing cycle occurs more frequently than the tax compliance cycle. It carries tax compliance information forward until the report is produced. The srt* utilities use this file for input only.

WARNING

The customer information is lost in the summation of the log.sum.

Remember to delete or move this file before running a utility that uses it, if a combined report is not desired.

5.4 General Tips When Using Utilities

AFC is designed to provide optimum flexibility in creating reports. The following tips have been accumulated to assist in making the best use of available options.

1. Combining Log Files
2. If an AFC log file exists when AFC is run, new tax data will be appended to the end of an existing log file (i.e. not placed into a new log file). Also, log files can be combined prior to passing to a utility, although they cannot be split apart after they are merged.
3. The criteria for deciding to combine files is based on how often AFC is run, how big the created log files are and the date cycles of your billing and tax compliance system. Note that most transactions will generate many tax records and that each tax record is either 100 or 168 bytes long, depending on the configuration.
4. Many clients use a new log file for each run of AFC to reduce the complexity introduced when a rerun or restart is required.
5. Most clients will do all of their customer billing processing within their Function programs and use the log file to process only tax compliance data at a later date.
6. Utilities are provided to merge two or more log files into a single log file. This is useful when running reports that reflect the combined data of the individual log files.
7. Primarily the srt* utilities will produce files sorted by the jurisdictional level.
8. AFC provides the asciilog.exe utility for maintenance of AFC log files, allowing the user to create an ASCII version of the log file.
9. Many utilities produce a log.sum file that accumulates tax compliance totals over multiple runs to generate a compilation of the tax totals.
10. The pkzip executable option listed with some utilities is only available with the windows platform.

5.5 AFC Utilities

Table 5-2 Utility Summary

Utility Name	Function	Short Description
asciilog.exe	File Management	Dumps the contents of the <i>logfilename.log</i> to an ASCII version of the log file
batchsplit.exe	Processing	Allows for two output log files. <i>logfilename1.log</i> is the voice data log and <i>logfilename2.log</i> is the data log.
commerge.exe	File Management	Combines the log file with the <i>log.sum</i> (if present) then sorts and combines the data.
comrp.exe	Reporting	Data formatted to produce a customer information file called <i>ComRPT.ssf</i> .
comptnum.exe	Reporting	Reads the log file specified in <i>filelocs.txt</i> and produces comma delimited text files <i>outputfilename.ssf</i> , <i>outputfilename.nca</i> , and <i>outputfilename.nba</i> (or a unified file <i>outputfilename.scl</i>) for tax compliance filing.
csf20.exe	Processing	Reads the log file specified in <i>filelocs.txt</i> . Uses a CDF filename from the command line to name the output files.
customsort.exe	Reporting	Produces a comma-delimited file for compliance filing. User selects sort criteria and preferences. Refer to the AFC Custom Sort Utility User Manual for more information on this utility.
ezcomprep.exe	Reporting	Produces a comma-delimited file for compliance filing. Sorting and combining is performed at the jurisdictional level.
ezlog_ns.exe	Reporting	Produces a comma-delimited file for compliance filing.
ezlogcust.exe	Reporting	Produces a comma-delimited file for customer billing. Sorting and reporting is done at the customer level.
ezlogcustios.exe	Reporting	Produces a comma-delimited file for customer billing. Sorting and reporting is done at the customer level.
ezlogcustpts.exe	Reporting	Produces a comma-delimited file for customer billing. Sorting and reporting is done at the customer level. The sort key also contains PCode, optional, and Service Level Number fields.
ezlogcustptsInl.exe	Reporting	Produces a comma-delimited file for customer billing. Sorting and reporting is done at the customer level. In addition to the PCode, optional, and Service Level Number fields and the number of lines are also included.
EZTax_20.exe	Processing	Refer to batchfile processing for details on the Batch processing method.
EZTaxappend.exe	File Management	Combines two log files.
EZTaxappendf.exe	File Management	Combines multiple log files.
log no tax transactions		This is an option in the EZTax.cfg file, placed here as it may effect the output.
srtcdf20.exe	Reporting	Produces a fixed length text file <i>logfilename.ssf</i> for tax compliance filing and a fixed length text file <i>filename.csf</i> for customer billing.
srtcdf20p.exe	Reporting	Produces a fixed length text file <i>EZTaxlogfilename.ssf</i> for tax compliance filing and a fixed length text file <i>logfilename.csf</i> for customer billing.
srtcomcust20.exe	Reporting	Produces a comma delimited text file <i>logfilename.ssf</i> for tax compliance filing and a fixed length text file <i>logfilename.csf</i> for customer billing.

Table 5-2 Utility Summary

Utility Name	Function	Short Description
srtcomma20.exe	Reporting	Produces a comma delimited text file <i>logfilename.ssf</i> for tax compliance filing and a fixed length text file <i>logfilename.csf</i> for customer billing.
srtcomma20l.exe	Reporting	Produces a comma delimited text file <i>logfilename.ssf</i> for tax compliance filing and a fixed length text file <i>logfilename.csf</i> for customer billing, separating adjustments. The .csf file is used for import into the billing system to populate customer bills. The comma-delimited .ssf file is sorted at the jurisdictional level. The .ssf file also contains the number of lines.
srtcomma20ld.exe	Reporting	Produces a comma delimited text file <i>logfilename.ssf</i> for tax compliance filing and a fixed length text file <i>logfilename.csf</i> for customer billing.
srtrev20l.exe	Reporting	Produces a comma delimited text file <i>logfilename.ssf</i> for tax compliance filing and a fixed length text file <i>logfilename.csf</i> for customer billing, separating adjustments. The .csf file is used for import into the billing system to populate customer bills. The comma-delimited .ssf file is sorted at the jurisdictional level. The .ssf file also contains the number of lines and the base sale amount. The base sale amount can be included in the csf by command-line option.
strg.exe	Reporting	Produces a fixed length text file <i>logfilename.ssf</i> for tax compliance filing. If the extra <i>logfilename</i> is specified, it is used for appending instead of the log.sum.
upsize_log.exe	File Management	Converts the short format <i>logfilename1</i> to a long format log <i>logfilename1</i> .

5.5.1 asciilog.exe

COMMAND LINE

```
asciilog logfilename.log <+p> <+tc> <+a> <+h> <-csv> <-?>
```

DESCRIPTION

The asciilog.exe command dumps the contents of the *logfilename.log* to an ASCII version of the log file. It does not sort or combine records. It produces a file named *logfilename.txt* which is a legible version of the *logfilename.log* data.

INPUT

Log file: *logfilename.log* file as defined in the command line and is NOT deleted at end of job.

OUTPUT

Comma delimited *logfilename.txt* or *logfilename.csv* file.

FILE FORMAT KEY

The contents of the file are highly dependent upon the command line arguments. Use the **+h** argument to prepend the output with a header row that defines each column.

ARGUMENTS (in any order)

- +p** option adds the P-Code to the output, located before the Country for each record
- +tc** option adds the tax category description to the end of each record
- +a** option outputs all available fields
- +h** option prepends the output with a header record
- csv** option uses .csv as the extension for easier import to other applications
- ?** option displays usage screen and exits

NOTES:

AFC log is NOT deleted.

Also, to have valid values for transaction type and service type, the AFC session that generated the log must have *logtransserv=true* (ref 7.2.3.8 Log Transaction / Service Types).

5.5.2 batchsplit.exe

COMMAND LINE

batchsplit *inputfilename.cdf* *logfilename1.log* *logfilename2.log* [-dD]

DESCRIPTION

The batchsplit command is a processing utility designed to accept records submitted in batch fashion. Records are supplied as an ASCII formatted file *inputfilename.cdf*. Two output log files are produced. *logfilename1.log* is the voice data log and *logfilename2.log* is the data log.

INPUT

Reads the *inputfilename.cdf* ASCII text file

OUTPUT

logfilename1 - as defined in the command line.

logfilename2 - as defined in the command line.

read_err.st - any errors found while editing input file are reported here.

billing.log - date/time stamp log (appended to).

bureau_log - run totals (appended to).

EZTax.sta - status

inputfilename.rpt - break down by totals (not for compliance filing).

FILE FORMAT KEY

Refer to Figure 6-2.

ARGUMENTS

logfilename1.log - Voice Data Log file name

logfilename2.log - Data Log file name

d or *D* - enables the default transaction/service type processing

NOTES:

If the *cust_no* field in the *cdf* file has an "I:" in it, the transaction will go to the data log. If not, the transaction will go to the voice log.

The *transactionservice.exp* file must be in the *EZtax* working directory.

EZTax.log is neither deleted nor created.

Filelocs.txt MUST be in the working directory.

SAMPLE DATA

678367N	704752N	678367NB20070109+00000005067	0	0000001000001SI	I:	6444U00000000013CDXXXC
678367N	408276N	678367NB20070109-14485457933	0	0000001000001SI		6444U00000003645DLFFFI
678367N	650786N	678367NB20070109+00000000600	0	0000001000001SI		6444U00000000002CDLXFC

5.5.3 commerge.exe

COMMAND LINE

commerge

DESCRIPTION

The commerge.exe command reads the log file specified in filelocs.txt, combines the log file with the log.sum (if present) then sorts and combines the data.

INPUT

Reads the log file as defined by filelocs.txt or AFC Initialization Function.
log.sum – optional input from previous run

OUTPUT

No output file is produced as this is a log manipulation utility.

FILE FORMAT KEY

The output file produced is a log file with no format key as this is a log manipulation utility.

ARGUMENTS

None.

NOTES:

The log.sum file is created

WARNING - EZTax.log is DELETED. Make backups in case reruns are needed.

SAMPLE DATA

No output file is produced as this is a log manipulation utility.

5.5.4 comptnum.exe

COMMAND LINE

```
comptnum [-z <pkzip executable>] outputfilename <extralogfilename> [-cl]
```

DESCRIPTION

The comptnum.exe command reads the log file specified in filelocs.txt and produces comma delimited text files *outputfilename.ssf*, *outputfilename.nca*, and *outputfilename.nba* (or a unified file *outputfilename.scl*) for tax compliance filing.

INPUT

Reads the *logfilename.log* file as defined by filelocs.txt or EZtax Initialization Function.
log.sum – optional input from previous run.

OUTPUT

Comma Delimited *outputfilename.ssf* (nca, nba, scl) file for tax compliance filing.
Sorted and condensed by PCode, Tax Type, Tax Level, Tax Rate

FILE FORMAT KEY

Country, State, County, Locality, PCode, Tax type, Tax Type Description, Tax level, Tax Level Description, Tax rate, Tax amount, Gross Sale, Exempt, Adjustments, Taxable measure, Minutes

Figure 5-3 comptnum SSF File Format Key	
Description	Type
Country	Alpha
State	Alpha
County	Alpha
Locality	Alpha
PCode	Numeric
Tax Type	Numeric
Tax Type Description	Alpha
Tax Level	Numeric
Tax Level Description	Alpha
Tax Rate	Numeric
Tax Amount	Numeric
Gross Sale	Numeric
Exempt	Numeric
Adjustments	Numeric
Taxable Measure	Numeric
Minutes	Numeric
Customer number	Alpha

ARGUMENTS

-z <pkzip executable> - path to zip executable file*
<output filename>
extra log file name – use instead of log.sum
-cl creates a combined compliance log file with a .scl extension, containing both the billable and non-billable amounts instead of a separate log file for each.

* The pkzip executable option is only available with the windows platform.

NOTES:

- The Tax Rate field is sensitive to positive and negative values
- The log.sum file is NOT created
- WARNING - EZTax.log is DELETED. Make backups in case reruns are needed.

SAMPLE DATA

USA, , , 0, 6, Federal Excise Tax, 0, Federal, 0.030000, 63.946941, 2131.564739, 0.000000, 0.000000, 2131.564739, 2376516.0, 1
USA, CA, , , 253500, 9, P.U.C. Fee, 1, State, 0.001100, 0.072776, 66.159998, 0.000000, 0.000000, 66.159998, 155916.0, 0000000000000002
USA, CA, , , 253500, 10, E911 Tax, 1, State, 0.006500, 0.450252, 69.269518, 0.000000, 0.000000, 69.269518, 155916.0, 0000000000004302
USA, CA, , , 253500, 19, State High Cost Fund, 1, State, 0.024300, 1.607688, 66.159998, 0.000000, 0.000000, 66.159998, 155916.0,
000000000099999
U, 1SA, CA, , , 253500, 21, CA Teleconnect Fund, 1, State, 0.001600, 0.105856, 66.159998, 0.000000, 0.000000, 66.159998,
155916.0, Customer Number 2
USA, CA, , , 253500, , 160, CA High Cost Fund A, 1, State, 0.001500, 0.099240, 66.159998, 0.000000, 0.000000, 66.159998,
155916.0, User value here
USA, CO, , , 427200, 13, State Universal Service Fund, 1, State, 0.029000, 0.026172, 0.902500, 0.000000, 0.000000, 0.902500, 1920.0,
xxxxxxxxxxxxxx2
USA, CO, DENVER, , 442100, 4, District Tax, 2, County, 0.001000, 0.000211, 0.210945, 0.000000, 0.000000, 0.210945, 246.0,
00000000000000000000

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

5.5.5 comrpt.exe

COMMAND LINE

comrpt [-cl]

DESCRIPTION

The comrpt.exe command reads the log.sum. The data is then formatted to produce a customer information file called *compliancessf*.

INPUT

Reads the log.sum file.

log.sum – optional input from previous run

OUTPUT

Fixed Length *compliancessf* file.

FILE FORMAT KEY

Figure 5-4 comrpt File Format Key			
Description	Type	Length	Positions
PCode	Numeric	9	1-9
Tax Type	Numeric	6	10-15
Tax Level	Numeric	1	16
Tax Rate	Numeric	8	17-24
Tax Amount	Numeric	12	25-36
File Record Length		36	

ARGUMENTS

-cl creates a combined compliance log file with a .scl extension, containing both the billable and non-billable amounts instead of a separate log file for each.

NOTES:

None.

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

5.5.6 csf20.exe

COMMAND LINE

`csf20 [-z <pkzip executable>] outputfilename [-cl]`

DESCRIPTION

The csf20.exe command is a processing utility that reads the log file specified in filelocs.txt. It produces a fixed length file with a .CSF extension. The command line argument determines the output file name.

INPUT

Reads the log file as defined by filelocs.txt.

OUTPUT

Fixed Length *outputfilename.csf* file.

FILE FORMAT KEY

Table 5-3 csf20.exe File Format Key			
Description	Type	Length	Positions
Customer Number	A/N	20	1-20
Tax type	A/N	6	21-26
Authority level	A/N	1	27
Tax amount sign	A	1	28
Tax amount	N	11.5	29-39
File record length		39	

ARGUMENTS

outputfilename

`-z <pkzip executable>` - path to zip executable file*

`-cl` creates a combined compliance log file with a .scl extension, containing both the billable and non-billable amounts instead of a separate log file for each.

* The pkzip executable option is only available with the windows platform.

NOTES:

EZTax.log is not deleted.

filelocs.txt MUST be in the working directory.

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

5.5.7 customsort.exe

COMMAND LINE

```
customsort -cfg customsort.cfg [-log logfile] [-data datapath] [-work workpath] [-ext extension] [-?]
```

DESCRIPTION

The custom sort utility is designed to allow clients to produce taxation reports which are sorted and summarized using client-specified fields and/or options. The utility is restricted to fields available in the AFC transaction log, which can be affected by configuration settings in the EZTax.cfg.

INPUT

Reads the log file as defined by filelocs.txt.

OUTPUT

User defined.

ARGUMENTS (in any order)

-cfg customsort.cfg	: Required. Custom sort configuration to be used.
-log logfile	: EZTax log file to be processed (DEF: EZTax.log).
-data <path>	: Data directory; overrides config file value.
-work <path>	: Working directory; overrides config file value.
-ext <ext>	: Result file extension without '.'; overrides reserved config file extension.
-summarize <sumfile>	: Log file is summarized into sumfile. On large log files, this option can substantially speed up the sorting process.
-out <fname>	: Output base filename; overrides config file. This option can be used when there is a need to set the output files from a script so that different sorts can share the same configuration file.
-?	: Display usage and exit

Refer to the **AFC Custom Sort Utility User Manual** for more information on this utility.

5.5.8 ezcomprep.exe

COMMAND LINE

ezcomprep [-cl]

DESCRIPTION

The ezcomprep.exe command reads the log file specified in filelocs.txt and produces a comma-delimited file for compliance filing. It does not include adjustments as part of Gross Sale. Sorting and combining is performed at the jurisdictional level. It combines the contents of log.sum if it exists.

INPUT

Reads the log file as defined by filelocs.txt or AFC Initialization Function.

log.sum – optional input from previous run

OUTPUT

log.sum – sorted and condensed log file as optional input for next run

Comma delimited *outputfilename.rpt* file.

Sorted and condensed by PCode, Tax Type, Tax Level, Tax Rate

FILE FORMAT KEY

Country, State, County, Locality, Tax type, Tax level, Tax rate, Tax amount, Gross Sale, Exempt, Adjustments, Taxable measure, Minutes

Figure 5-5 ezcomprep File Format Key	
Description	Type
Country	Alpha
State	Alpha
County	Alpha
Locality	Alpha
Tax Type	Numeric
Tax Level	Numeric
Tax Rate	Numeric
Tax Amount	Numeric
Gross Sale	Numeric
Exempt	Numeric
Adjustments	Numeric
Taxable Measure	Numeric
Minutes	Numeric

ARGUMENTS

`-cl` creates a combined compliance log file with a .scl extension, containing both the billable and non-billable amounts instead of a separate log file for each.

NOTES:

WARNING - EZTax.log is DELETED. Make backups in case reruns are needed.

SAMPLE DATA

USA,,, 6, 0, 0.030000, 29881.866905, 996062.252444, 0.000000, 0.000000, 996062.252444, 14859213.000000

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

5.5.9 ezlog_ns.exe

COMMAND LINE

`ezlog_ns outputfilename [-cl]`

DESCRIPTION

The `ezlog_ns.exe` command reads the log file specified in `filelocs.txt` and produces a comma-delimited file for compliance filing.

INPUT

Reads the log file name from `filelocs.txt`.

`log.sum` – optional input from previous run

OUTPUT

`log.sum` – sorted and condensed log file as optional input for next run

Comma delimited `outputfilename.ssf` file.

Sorted and condensed by PCode, Tax Type, Tax Level, Tax Rate

FILE FORMAT KEY

Country, State, County, Locality, Tax type, Tax level, Tax rate, Tax amount, Gross Sale, Exempt, Adjustments, Taxable measure, Minutes

Figure 5-6 ezlog_ns File Format Key	
Description	Type
Country	Alpha
State	Alpha
County	Alpha
Locality	Alpha
Tax Type	Numeric
Tax Level	Numeric
Tax Rate	Numeric
Tax Amount	Numeric
Gross Sale	Numeric
Exempt	Numeric
Adjustments	Numeric
Taxable Measure	Numeric
Minutes	Numeric

ARGUMENTS

`outputfilename`

`-cl` creates a combined compliance log file with a `.scl` extension, containing both the billable and non-billable amounts instead of a separate log file for each.

NOTES:

WARNING - EZTax.log is DELETED. Make backups in case reruns are needed.

SAMPLE DATA

USA,,, 6, 0, 0.030000, 29881.866905, 996062.252444, 0.000000, 0.000000, 996062.252444, 14859213.000000

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

5.5.10 ezlogcust.exe

COMMAND LINE

ezlogcust [-cl]

DESCRIPTION

The ezlogcust.exe command reads the log file specified in filelocs.txt and produces a comma-delimited file for customer billing. Sorting and reporting is done at the customer level.

INPUT

Reads the log file as defined by filelocs.txt or AFC Initialization Function.
log.sum – optional input from previous run

OUTPUT

log.sum – sorted and condensed log file as optional input for next run
Comma delimited log.rpt file.
Sorted and condensed by Customer Number, PCode, Tax Type, Tax Level, Tax Rate, Calculation Type.

FILE FORMAT KEY

Country, State, County, Locality, Tax type, Tax level, Tax rate, Calculation Type, Tax amount, Gross Sale, Exempt, Adjustments, Taxable measure, Minutes, Lines, Customer Number

Figure 5-7 ezlogcust File Format Key	
Description	Type
Country	Alpha
State	Alpha
County	Alpha
Locality	Alpha
Tax Type	Numeric
Tax Level	Numeric
Tax Rate	Numeric
Calculation Type	Numeric
Tax Amount	Numeric
Gross Sale	Numeric
Exempt	Numeric
Adjustments	Numeric
Taxable Measure	Numeric
Minutes	Numeric
Lines	Numeric
Customer Number	Alpha Numeric

ARGUMENTS

-cl creates a combined compliance log file with a .scl extension, containing both the billable and non-billable amounts instead of a separate log file for each.

NOTES:

WARNING - EZTax.log is DELETED. Make backups in case reruns are needed.

SAMPLE DATA

```
USA, ,,, 6, 0, 0.030000, RATE, 0.892576, 29.752527, 0.000000, 0.000000, 29.752527, 0.000000, 1, 1, 2143278889
USA, TX, ,,, 9, 1, 0.001670, RATE, 0.049687, 29.752527, 0.000000, 0.000000, 29.752527, 0.000000, 1, 0, 2143278889
USA, TX, ,,, 13, 1, 0.036000, RATE, 1.035438, 28.762179, 0.000000, 0.000000, 28.762179, 0.000000, 1, 0, 2143278889
USA, TX, ,,, 26, 1, 0.012500, RATE, 0.367089, 29.367090, 0.000000, 0.000000, 29.367090, 0.000000, 1, 0, 2143278889
USA, TX, DALLAS, DALLAS, 1, 1, 0.062500, RATE, 1.863693, 29.819089, 0.000000, 0.000000, 29.819089, 0.000000, 1, 0, 2143278889
USA, TX, DALLAS, DALLAS, 1, 3, 0.010000, RATE, 0.298022, 29.802214, 0.000000, 0.000000, 29.802214, 0.000000, 1, 0, 2143278889
USA, TX, DALLAS, DALLAS, 4, 3, 0.010000, RATE, 0.283838, 28.383766, 0.000000, 0.000000, 28.383766, 0.000000, 0, 0, 2143278889
USA, TX, DALLAS, DALLAS, 10, 3, 0.620000, PER_LINE, 0.620000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 1, 0, 2143278889
USA, TX, DALLAS, DALLAS, 51, 3, 1.350000, PER_LINE, 1.350000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 1, 0, 2143278889
USA, ,,, 6, 0, 0.030000, RATE, 0.210216, 7.007192, 0.000000, 0.000000, 7.007192, 5.100000, 0, 0, 2143278889B
USA, ,,, 18, 0, 0.072805, RATE, 0.009967, 0.136896, 0.000000, 0.000000, 0.136896, 0.100000, 0, 0, 2143278889B
USA, ,,, 31, 0, 0.000730, RATE, 0.000099, 0.135089, 0.000000, 0.000000, 0.135089, 0.100000, 0, 0, 2143278889B
USA, TX, ,,, 9, 1, 0.001670, RATE, 0.011392, 6.821736, 0.000000, 0.000000, 6.821736, 5.000000, 0, 0, 2143278889B
USA, TX, ,,, 10, 1, 0.006000, RATE, 0.039000, 6.500000, 0.000000, 0.000000, 6.500000, 5.000000, 0, 0, 2143278889B
USA, TX, ,,, 13, 1, 0.036000, RATE, 0.242515, 6.736532, 0.000000, 0.000000, 6.736532, 5.100000, 0, 0, 2143278889B
USA, TX, ,,, 26, 1, 0.012500, RATE, 0.086118, 6.889438, 0.000000, 0.000000, 6.889438, 5.100000, 0, 0, 2143278889B
USA, TX, DALLAS, DALLAS, 1, 1, 0.062500, RATE, 0.436224, 6.979585, 0.000000, 0.000000, 6.979585, 5.100000, 0, 0, 2143278889B
USA, TX, DALLAS, DALLAS, 1, 3, 0.010000, RATE, 0.068331, 6.833129, 0.000000, 0.000000, 6.833129, 5.000000, 0, 0, 2143278889B
USA, TX, DALLAS, DALLAS, 4, 3, 0.010000, RATE, 0.068331, 6.833129, 0.000000, 0.000000, 6.833129, 5.000000, 0, 0, 2143278889B
```

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

5.5.11 ezlogcustios.exe

COMMAND LINE

ezlogcustios [-cl]

DESCRIPTION

The ezlogcustios.exe command reads the log file specified in filelocs.txt and produces a comma-delimited file for customer billing. Sorting and reporting is done at the customer level.

INPUT

Reads the log file specified in filelocs.txt or AFC Initialization Function.
log.sum – optional input from previous run

OUTPUT

log.sum – sorted and condensed log file as optional input for next run
Comma delimited log.rpt file.
Sorted and condensed by Customer Number, PCode, Invoice Number, Optional, Service Level Number, Tax Type, Tax Level, Tax Rate

FILE FORMAT KEY

Invoice Number, Optional, Service Level Number, Country, State, County, Locality, Tax type, Tax level, Tax rate, Tax amount, Gross Sale, Exempt, Adjustments, Taxable measure, Minutes, Customer Number

Figure 5-8 ezlogcustios File Format Key

Description	Type
Invoice Number	Numeric
Optional	Numeric
Service Level Number	Numeric
Country	Alpha
State	Alpha
County	Alpha
Locality	Alpha
Tax Type	Numeric
Tax Level	Numeric
Tax Rate	Numeric
Tax Amount	Numeric
Gross Sale	Numeric
Exempt	Numeric
Adjustments	Numeric
Taxable Measure	Numeric
Minutes	Numeric

Figure 5-8 ezlogcustios File Format Key

Description	Type
Customer Number	Alpha Numeric

ARGUMENTS

-cl creates a combined compliance log file with a .scl extension, containing both the billable and non-billable amounts instead of a separate log file for each.

NOTES:

WARNING - EZTax.log is DELETED. Make backups in case reruns are needed.

SAMPLE DATA

6, 8, 7, USA, AL, AUTAUGA, AUTAUGAVILLE, 1, 1, 0.040000, RATE, 0.720000, 18.000000, 0.000000, 0.000000, 18.000000, 275.940002, 54, 18, BillSoft Inc.
6, 8, 7, USA, AL, AUTAUGA, JONES, 1, 2, 0.020000, RATE, 0.320000, 16.000000, 0.000000, 0.000000, 16.000000, 245.280014, 48, 16, BillSoft Inc.
6, 8, 7, USA, AL, AUTAUGA, PRATTVILLE, 1, 3, 0.025000, RATE, 0.050000, 2.000000, 0.000000, 0.000000, 2.000000, 30.660000, 6, 2, BillSoft Inc.
6, 8, 7, USA, AL, AUTAUGA, AUTAUGAVILLE, 1, 3, 0.030000, RATE, 0.060000, 2.000000, 0.000000, 0.000000, 2.000000, 30.660000, 6, 2, BillSoft Inc.
6, 8, 7, USA, , , 6, 0, 0.030000, RATE, 14.312928, 477.097605, 0.000000, 0.000000, 477.097605, 5825.416504, 1140, 380, BillSoft Inc.
6, 8, 7, USA, AL, , , 8, 1, 0.060000, RATE, 2.200486, 36.674766, 0.000000, 0.000000, 36.674766, 551.879822, 108, 36, BillSoft Inc.
6, 8, 7, USA, AL, BALDWIN, , 10, 2, 0.690000, PER_LINE, 4.140000, 2.000000, 0.000000, 0.000000, 2.000000, 30.660000, 6, 2, BillSoft Inc.
6, 8, 7, USA, AL, AUTAUGA, , 10, 2, 0.850000, PER_LINE, 35.700001, 14.000000, 0.000000, 0.000000, 14.000000, 214.620010, 42, 14, BillSoft Inc.
6, 8, 7, USA, AL, , , 16, 1, 0.060000, RATE, 7.759929, 129.332160, 0.000000, 0.000000, 129.332160, 1931.577759, 378, 126, BillSoft Inc.
6, 8, 7, USA, , , 18, 0, 0.089000, RATE, 5.340000, 60.000000, 0.000000, 0.000000, 60.000000, 919.800232, 180, 60, BillSoft Inc.
6, 8, 7, USA, AL, , , 23, 1, 0.150000, PER_LINE, 16.200001, 36.000000, 0.000000, 0.000000, 36.000000, 551.879822, 108, 36, BillSoft Inc.
6, 8, 7, USA, , , 31, 0, 0.003560, RATE, 0.213600, 60.000000, 0.000000, 0.000000, 60.000000, 919.800232, 180, 60, BillSoft Inc.
6, 8, 7, USA, AL, BALDWIN, , 35, 2, 1.950000, PER_LINE, 11.700000, 2.000000, 0.000000, 0.000000, 2.000000, 30.660000, 6, 2, BillSoft Inc.
6, 8, 7, USA, AL, AUTAUGA, , 35, 2, 2.400000, PER_LINE, 100.800004, 14.000000, 0.000000, 0.000000, 14.000000, 214.620010, 42, 14, BillSoft Inc.
6, 8, 7, USA, AL, , , 44, 1, 0.700000, FIXED, 12.600000, 18.474840, 0.000000, 0.000000, 18.474840, 275.940002, 54, 18, BillSoft Inc.
6, 8, 7, USA, , , 55, 0, 0.025365, RATE, 1.014600, 40.000000, 0.000000, 0.000000, 40.000000, 613.199890, 120, 40, BillSoft Inc.
6, 8, 7, USA, , , 56, 0, 0.010680, RATE, 0.213600, 20.000000, 0.000000, 0.000000, 20.000000, 306.599976, 60, 20, BillSoft Inc.
6, 8, 7, USA, , , 62, 0, 0.001015, RATE, 0.040600, 40.000000, 0.000000, 0.000000, 40.000000, 613.199890, 120, 40, BillSoft Inc.
6, 8, 7, USA, , , 63, 0, 0.000427, RATE, 0.008540, 20.000000, 0.000000, 0.000000, 20.000000, 306.599976, 60, 20, BillSoft Inc.

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

5.5.12 ezlogcustpts.exe

COMMAND LINE

ezlogcustpts [-cl]

DESCRIPTION

The ezlogcustpts.exe command reads the log file specified in filelocs.txt and produces a comma-delimited file for customer billing. Sorting and reporting is done at the customer level. The sort key also contains PCode, optional, and Service Level Number fields.

INPUT

Reads the log file as defined by filelocs.txt or AFC Initialization Function.

log.sum – optional input from previous run

OUTPUT

log.sum – sorted and condensed log file as optional input for next run

Comma delimited log.rpt file.

Sorted and condensed by Customer Number, PCode, Optional, Service Level Number, Tax Type, Tax Level, Tax Rate

FILE FORMAT KEY

PCode, Transaction type, Service Type, Country, State, County, Locality, Tax type, Tax level, Tax rate, Tax amount, Gross Sale, Exempt, Adjustments, Taxable measure, Minutes, Customer Number.

Figure 5-9 ezlogcustpts File Format Key

Description	Type
PCode	Numeric
Optional	Numeric
Service Level Number	Numeric
Country	Alpha
State	Alpha
County	Alpha
Locality	Alpha
Tax Type	Numeric
Tax Level	Numeric
Tax Rate	Numeric
Tax Amount	Numeric
Gross Sale	Numeric
Exempt	Numeric
Adjustments	Numeric
Taxable Measure	Numeric
Minutes	Numeric

Figure 5-9 ezlogcustpts File Format Key

Description	Type
Customer Number	Alpha Numeric

ARGUMENTS

-cl creates a combined compliance log file with a .scl extension, containing both the billable and non-billable amounts instead of a separate log file for each.

NOTES:

- **WARNING - EZTax.log is DELETED.** Make backups in case reruns are needed.

SAMPLE DATA

```
0, 1, 1, USA,,, 6, 0, 0.030000, 0.128824, 4.294140, 0.000000, 0.000000, 4.294140, 61.320000, BillSoft, Inc.  
0, 1, 1, USA,,, 18, 0, 0.072805, 0.291220, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
0, 1, 1, USA,,, 31, 0, 0.000730, 0.002920, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
0, 1, 2, USA,,, 6, 0, 0.030000, 0.128824, 4.294140, 0.000000, 0.000000, 4.294140, 61.320000, BillSoft, Inc.  
0, 1, 2, USA,,, 18, 0, 0.072805, 0.291220, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
0, 1, 2, USA,,, 31, 0, 0.000730, 0.002920, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
0, 1, 3, USA,,, 6, 0, 0.030000, 0.128824, 4.294140, 0.000000, 0.000000, 4.294140, 61.320000, BillSoft, Inc.  
0, 1, 3, USA,,, 18, 0, 0.072805, 0.291220, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
0, 1, 3, USA,,, 31, 0, 0.000730, 0.002920, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
0, 1, 4, USA,,, 18, 0, 0.072805, 0.291220, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
0, 1, 4, USA,,, 31, 0, 0.000730, 0.002920, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
0, 2, 1, USA,,, 6, 0, 0.030000, 0.120000, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
0, 2, 2, USA,,, 6, 0, 0.030000, 0.120000, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
0, 2, 3, USA,,, 6, 0, 0.030000, 0.120000, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
0, 2, 5, USA,,, 6, 0, 0.030000, 0.120000, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
0, 3, 6, USA,,, 6, 0, 0.030000, 0.360000, 12.000000, 0.000000, 0.000000, 12.000000, 183.960007, BillSoft, Inc.  
0, 3, 9, USA,,, 6, 0, 0.030000, 0.120000, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
100, 1, 1, USA, AL, , , 16, 1, 0.060000, 0.257648, 4.294140, 0.000000, 0.000000, 4.294140, 61.320000, BillSoft, Inc.  
100, 1, 2, USA, AL, , , 16, 1, 0.060000, 0.257648, 4.294140, 0.000000, 0.000000, 4.294140, 61.320000, BillSoft, Inc.  
100, 1, 3, USA, AL, , , 16, 1, 0.060000, 0.257648, 4.294140, 0.000000, 0.000000, 4.294140, 61.320000, BillSoft, Inc.  
100, 1, 4, USA, AL, , , 16, 1, 0.060000, 0.257648, 4.294140, 0.000000, 0.000000, 4.294140, 61.320000, BillSoft, Inc.  
100, 2, 1, USA, AL, , , 16, 1, 0.060000, 0.240000, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
100, 2, 2, USA, AL, , , 16, 1, 0.060000, 0.240000, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
100, 2, 3, USA, AL, , , 16, 1, 0.060000, 0.240000, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
100, 2, 4, USA, AL, , , 16, 1, 0.060000, 0.240000, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
100, 2, 5, USA, AL, , , 16, 1, 0.060000, 0.240000, 4.000000, 0.000000, 0.000000, 4.000000, 61.320000, BillSoft, Inc.  
100, 3, 6, USA, AL, , , 16, 1, 0.060000, 0.720000, 12.000000, 0.000000, 0.000000, 12.000000, 183.960007, BillSoft, Inc.  
200, 2, 5, USA, AL, AUTAUGA, , 10, 2, 0.050000, 0.100000, 2.000000, 0.000000, 0.000000, 2.000000, 30.660000, BillSoft, Inc.
```

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

5.5.13 ezlogcustptslnl.exe

COMMAND LINE

ezlogcustptslnl [-cl]

DESCRIPTION

The ezlogcustptslnl.exe command reads the log file specified in filelocs.txt and produces a comma-delimited file for customer billing. Sorting and reporting is done at the customer level. In addition to the PCode, optional, and Service Level Number fields and the number of lines and are also included.

INPUT

Reads the log file as defined by filelocs.txt or AFC Initialization Function.
log.sum – optional input from previous run

OUTPUT

log.sum – sorted and condensed log file as optional input for next run
Comma delimited log.rpt file.

Sorted and condensed by Customer Number, PCode, Optional, Service Level Number, Tax Type, Tax Level, Tax Rate

FILE FORMAT KEY

PCode, Transaction type, Service Type, Country, State, County, Locality, Tax type, Tax level, Tax rate, Tax amount, Gross Sale, Exempt, Adjustments, Taxable measure, Minutes, Customer Number, Lines

Figure 5-10 ezlogcustptslnl File Format Key	
Description	Type
PCode	Numeric
Optional	Numeric
Service Level Number	Numeric
Country	Alpha
State	Alpha
County	Alpha
Locality	Alpha
Tax Type	Numeric
Tax Level	Numeric
Tax Rate	Numeric
Tax Amount	Numeric
Gross Sale	Numeric
Exempt	Numeric
Adjustments	Numeric

Figure 5-10 ezlogcustptslnl File Format Key	
Description	Type
Taxable Measure	Numeric
Minutes	Numeric
Customer Number	Alpha Numeric
Lines	Numeric

ARGUMENTS

-cl creates a combined compliance log file with a .scl extension, containing both the billable and non-billable amounts instead of a separate log file for each.

NOTES:

- WARNING - EZTax.log is DELETED. Make backups in case reruns are needed.

SAMPLE DATA

```

0, 0, 0, USA, , , 6, 0, 0.030000, 2.462930, 82.097663, 0.000000, 0.000000, 82.097663, 0.000000, BillSoft, Inc. , 6, 2
0, 0, 0, USA, , , 18, 0, 0.089000, 1.157000, 13.000000, 0.000000, 0.000000, 13.000000, 0.000000, BillSoft, Inc. , 0, 0
0, 0, 0, USA, , , 31, 0, 0.000000, 0.000000, 13.000000, 0.000000, 0.000000, 0.000000, 0.000000, BillSoft, Inc. , 0, 0
1210800, 0, 0, USA, KS, , 13, 1, 0.048700, 1.458565, 29.949999, 0.000000, 0.000000, 29.949999, 0.000000, BillSoft, Inc. , 0, 0
1284800, 0, 0, USA, KS, SALINE, , 10, 2, 0.750000, 2.250000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, BillSoft, Inc. , 3, 0
1284900, 0, 0, USA, KS, SALINE, SALINA, 1, 1, 0.053000, 2.039814, 38.487064, 0.000000, 0.000000, 38.487064, 0.000000, BillSoft, Inc. , 0, 0
1284900, 0, 0, USA, KS, SALINE, SALINA, 1, 2, 0.010000, 0.384871, 38.487064, 0.000000, 0.000000, 38.487064, 0.000000, BillSoft, Inc. , 0, 0
1284900, 0, 0, USA, KS, SALINE, SALINA, 1, 3, 0.007500, 0.288653, 38.487064, 0.000000, 0.000000, 38.487064, 0.000000, BillSoft, Inc. , 0, 0
2502500, 0, 0, USA, NY, , 5, 1, 0.025000, 0.925712, 37.028499, 0.000000, 0.000000, 37.028499, 0.000000, BillSoft, Inc. , 0, 0
2502500, 0, 0, USA, NY, , 45, 1, 0.003750, 0.138857, 37.028499, 0.000000, 0.000000, 37.028499, 0.000000, BillSoft, Inc. , 0, 0
2604000, 0, 0, USA, NY, NEW YORK, , 10, 2, 1.000000, 3.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, BillSoft, Inc. , 3, 0
2604000, 0, 0, USA, NY, NEW YORK, , 27, 2, 0.005950, 0.220320, 37.028499, 0.000000, 0.000000, 37.028499, 0.000000, BillSoft, Inc. , 0, 0
2604000, 0, 0, USA, NY, NEW YORK, , 28, 2, 0.001275, 0.047211, 37.028499, 0.000000, 0.000000, 37.028499, 0.000000, BillSoft, Inc. , 0, 0
2604100, 0, 0, USA, NY, NEW YORK, NEW YORK, 1, 1, 0.042500, 1.630325, 38.360599, 0.000000, 0.000000, 38.360599, 0.000000, BillSoft, Inc. , 0, 0
2604100, 0, 0, USA, NY, NEW YORK, NEW YORK, 1, 3, 0.043750, 1.678276, 38.360599, 0.000000, 0.000000, 38.360599, 0.000000, BillSoft, Inc. , 0, 0
2604100, 0, 0, USA, NY, NEW YORK, NEW YORK, 29, 3, 0.023500, 0.870170, 37.028499, 0.000000, 0.000000, 37.028499, 0.000000, BillSoft, Inc. , 0, 0

```

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

[5.5.14 EZTax_20.exe](#)

EZTax_20 is a Processing Utility used to pass transaction records through the AFC Engine in Batch fashion. Refer to batch file processing for details on the Batch processing method.

[5.5.15 EZTaxappend.exe](#)

COMMAND LINE

`EZTaxappend logfilename1.log logfilename2.log`

DESCRIPTION

The EZTaxappend.exe command is used to combine two *logfilename.log* files. It appends *logfilename1.log* to *logfilename2.log*, leaving *logfilename1.log* intact.

INPUT

logfilename1.log

OUTPUT

logfilename2.log

FILE FORMAT KEY

The output file produced is a log file with no format key as this is a log manipulation utility.

ARGUMENTS

logfilename1.log – the name of the EZTaxlog file to be appended
logfilename2.log – the name of the EZTaxlog file to be appended to.

NOTES:

- *log.sum* is NOT created.
- EZTax.log is NOT deleted.
- *logfilename1.log* and *logfilename2.log* must be actual filenames.
- Filelocs.txt is not used.

5.5.16 EZTaxappendf.exe

COMMAND LINE

EZTaxappendf *filelist.txt extaxlogfilename.log*

DESCRIPTION

The EZTaxappendf.exe command is used to combine the *logfilename.log* files listed in the *filelist.txt* text file containing the paths and *logfilename.log* names to combine.

INPUT

Filelist.txt

OUTPUT

logfilename.log

FILE FORMAT KEY

The output file produced is a log file with no format key as this is a log manipulation utility.

ARGUMENTS

Filelist.txt – the text file supplied by the user with the names of the *logfilename.log* files.

logfilename.log – the name of the new *logfilename.log* to be created.

NOTES:

- *logfilename.log* will be created if it does not exist. Otherwise it will append to the existing *logfilename.log* file.
- All log files defined in *Filelist.txt* are concatenated to *logfilename.log*
- *log.sum* is NOT created.
- EZTax.log is NOT deleted.
- Filelocs.txt is not used.
- *Filelist.txt* entries can be space or line delimited.

5.5.17 log no tax transactions

DESCRIPTION

LOGNOTAXTRANS=OFF
LOGNOTAXTRANS=ON

The Log No Tax Transaction option is used to allow writing zero tax amount transactions to the EZTax.ntl log file. The default setting of “OFF” disables this logging function. When set to “ON,” AFC will write zero tax amount transactions to the EZTax.ntl log file.

The no tax transaction log is used to track transactions that do not return any tax types and therefore do not perform tax calculations. This allows for the distinction from a transaction that logs zero tax when a calculation has occurred and resulted in a value of zero which is then placed in EZTax.log.

The no tax transaction log will be named **[logname].ntl** in which *logname* refers to the name of the AFC transaction log file, minus the extension, and will be output to the same directory as the log file.

Example of Naming Convention:

(Based on the assumption that the billing system has two sessions with log files named SessionA.log and SessionB.log.)

The output from a normal run with no tax transaction logging enabled would be:

SessionA.log
SessionA.sta
SessionA.tsr
SessionA.ntl
SessionB.log
SessionB.sta
SessionB.tsr
SessionB.ntl

NOTE:

The lognotaxtrans switch is over-ridden by the overall logging switch. If logging is turned off when the EZTaxInitEx function initializes the AFC session, the no-tax-trans log will not be created.

FILE FORMAT KEY

Short Format:

Company Identifier, Customer Number, Transaction Type, Service Type, P Code, Charge

Figure 5-11 Short EZTax.ntl Log File Format Key			
Description	Type	Length	Positions
Company Identifier	Alpha Numeric	20	1-20
Customer Number	Alpha Numeric	20	21-40
Transaction Type	Numeric	4	41-44
Service Type	Numeric	4	45-48
P Code	Numeric	10	49-58
Charge	Numeric	15.5	59-73
Record Length		73	

Long Format:

Company Identifier, Customer Number, Optional Alpha Text, Transaction Type, Service Type, P Code, Charge

Figure 5-12 EZTax.ntl Long Log File Format Key			
Description	Type	Length	Positions
Company Identifier	Alpha Numeric	20	1-20
Customer Number	Alpha Numeric	20	21-40
Optional Alpha	Alpha	20	41-60
Transaction Type	Numeric	4	61-64
Service Type	Numeric	4	65-68
P Code	Numeric	10	69-78
Charge	Numeric	15.5	79-93
Record Length		93	

NOTES:

None

SAMPLE DATA

Short Format:

company_identifier, customer_number, transaction_type, service_type, P_Code, charge

--- Sample output begins on the next line -----

EZtax version 9.0.0.0.5 No-tax Transaction Log

6215687423a,	TU98602746, 0001, 0027,	2102300,	1395.00000
6215687423a,	OP97602744, 0001, 0027,	2102300,	630.00000
6215687423a,	104602742, 0004, 0008,	2102300,	950.00000
6215687423a,	555602718, 0001, 0027,	2102300,	100.00000
6215687423a,	16055602716, 0001, 0027,	2102300,	100.00000
6215687423a,	22602711, 0001, 0027,	2102300,	100.00000
6215687423a,	19602709, 0003, 0006,	2102300,	266.44000
6215687423a,	33602707, 0001, 0027,	2102300,	300.00000
6215687423a,	55602705, 0001, 0027,	2102300,	500.00000
6215687423a,	LD55602703, 0001, 0027,	2102300,	100.00000
6215687423a,	56602701, 0003, 0006,	2102300,	266.44000
6215687423a,	57602699, 0001, 0027,	2102300,	300.00000
6215687423a,	58602697, 0001, 0027,	2102300,	500.00000
6215687423a,	58587819, 0001, 0027,	2350600,	500.00000

Long Format:

company_identifier, customer_number, optional alpha text, transaction_type, service_type, P_Code, charge

--- Sample output begins on the next line -----

EZtax version 9.0.0.0.5 No-tax Transaction Log

6215687423a,	TU98602746,	Optional Text, 0001, 0027,	2102300,	1395.00000
6215687423a,	OP97602744,	Optional Text, 0001, 0027,	2102300,	630.00000
6215687423a,	104602742,	Optional Text, 0004, 0008,	2102300,	950.00000
6215687423a,	555602718,	Optional Text, 0001, 0027,	2102300,	100.00000
6215687423a,	16055602716,	Optional Text, 0001, 0027,	2102300,	100.00000
6215687423a,	22602711,	Optional Text, 0001, 0027,	2102300,	100.00000
6215687423a,	19602709,	Optional Text, 0003, 0006,	2102300,	266.44000
6215687423a,	33602707,	Optional Text, 0001, 0027,	2102300,	300.00000
6215687423a,	55602705,	Optional Text, 0001, 0027,	2102300,	500.00000
6215687423a,	LD55602703,	Optional Text, 0001, 0027,	2102300,	100.00000
6215687423a,	56602701,	Optional Text, 0003, 0006,	2102300,	266.44000
6215687423a,	57602699,	Optional Text, 0001, 0027,	2102300,	300.00000
6215687423a,	58602697,	Optional Text, 0001, 0027,	2102300,	500.00000
6215687423a,	58587819,	Optional Text, 0001, 0027,	2350600,	500.00000

5.5.18 srtcdf20.exe

COMMAND LINE

srtcdf20 [-z <pkzip executable>] *outputfilename* [-n*N*] [*extralogfilename*] [-cl]

DESCRIPTION

The srtcdf20.exe command reads the log file specified in filelocs.txt and produces a fixed length text file *outputfilename.ssf* for tax compliance filing and a fixed length text file *outputfilename.csf* for customer billing.

INPUT

Reads the log file as defined by filelocs.txt or AFC Initialization Function.

log.sum – optional input from previous run, or uses the *extralogfilename* from the command line.

OUTPUT

Fixed Length *outputfilename.ssf* file for tax compliance filing.

Sorted and condensed by PCode, Tax Type, Tax Level, Tax Rate

Fixed Length *outputfilename.csf* file for customer billing.

Sorted and condensed by Customer Number, Tax Type, Tax Level, Tax Rate

FILE FORMAT KEYS

Figure 5-13 srtcdf20 SSF File Format Key

Description	Type	Length	Positions
PCode	Numeric	9	1-9
Tax Type	Numeric	6	10-15
Tax Level	Numeric	1	16
Tax Rate	Numeric	8.5	17-24
Tax Amount Sign	Alpha	1	25
Tax Amount	Numeric	11.5	26-36
File Record Length		36	

Figure 5-14 srtcdf20 CSF File Format Key

Description	Type	Length	Positions
Customer Number	Numeric	20	1-20
Tax Type	Numeric	6	21-26
Tax Level	Numeric	1	27
Tax Amount	Numeric	12	28-39
File Record Length		39	

ARGUMENTS:

-z <pkzip executable> - path to zip executable file*
N or n – CSF file is not sorted
extra log file name – use instead of log.sum
-cl creates a combined compliance log file with a .scl extension, containing both the billable and non-billable amounts instead of a separate log file for each.

* The pkzip executable option is only available with the windows platform.

NOTES:

- The Tax Rate field is sensitive to positive and negative values
- The log.sum file is NOT created
- WARNING - EZTax.log is DELETED. Make backups in case reruns are needed.

SAMPLE DATA

0	1	2	3
123456789012345678901234567890123456			

0000000000000006000003000+00098418707			
0000000000000007000000580+00018387113			
0000000000000018000003180+00081599822			
000000000000031000000039+00001000752			

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

5.5.19 srtcdf20p.exe

COMMAND LINE

srtcdf20p [-z <pkzippath>] *outputfilename* [-nN] [-cl]

DESCRIPTION

The srtcdf20p.exe command reads the log file specified in filelocs.txt and produces a fixed length text file *outputfilename.ssf* for tax compliance filing and a fixed length text file *outputfilename.csf* for customer billing.

The .csf file is used for import into the billing system to populate customer bills. Various compliance vendors use the .ssf file. The csf file also contains the PCode.

INPUT

Reads the logfile as defined by filelocs.txt or AFC Initialization Function.
log.sum – optional input from previous run

OUTPUT

Fixed Length *outputfilename.ssf* file for tax compliance filing.
Sorted and condensed by PCode, Tax Type, Tax Level, Tax Rate

Fixed Length *outputfilename.csf* file for customer billing.
Sorted and condensed by Customer Number, Tax Type, Tax Level, Tax Rate

FILE FORMAT KEYS

Figure 5-15 srtcdf20p SSF File Format Key			
Description	Type	Length	Positions
PCode	Numeric	9	1-9
Tax Type	Numeric	6	10-15
Tax Level	Numeric	1	16
Tax Rate	Numeric	8.5	17-24
Tax Amount Sign	Alpha	1	25
Tax Amount	Numeric	11.5	26-36
File Record Length		36	

Figure 5-16 srtcdf20p CSF File Format Key			
Description	Type	Length	Positions
Customer Number	Numeric	20	1-20
Tax Type	Numeric	6	21-26
Tax Level	Numeric	1	27
Tax Amount	Numeric	12	28-39
PCode	Numeric	9	40-48

Figure 5-16 srtcdf20p CSF File Format Key			
Description	Type	Length	Positions
File Record Length		48	

ARGUMENTS:

- z <pkzip executable> - path to zip executable file*
- N or n – CSF file is not sorted
- cl creates a combined compliance log file with a .scl extension, containing both the billable and non-billable amounts instead of a separate log file for each.

* The pkzip executable option is only available with the windows platform.

NOTES:

- The Tax Rate field is sensitive to positive and negative values
- The log.sum file is NOT created
- WARNING - EZTax.log is DELETED. Make backups in case reruns are needed.
-

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

5.5.20 srtcomcust20.exe

COMMAND LINE

srtcomcust20 [-z <pkzip executable>] *outputfilename [nN]* [*extra log file name*] [-cl]

DESCRIPTION

The srtcomcust20.exe command reads the log file specified in filelocs.txt and produces a comma delimited text file *outputfilename.ssf* for tax compliance filing and a fixed length text file *outputfilename.csf* for customer billing.

The .csf file is used for import into the billing system to populate customer bills. The comma-delimited .ssf file is sorted at the customer level.

INPUT

Reads the log file as defined by filelocs.txt or EZtax Initialization Function.

log.sum – optional input from previous run or uses the extra log file name from the command line.

OUTPUT

Comma Delimited *outputfilename.ssf* file for tax compliance filing.

Sorted and condensed by Customer Number, PCode, Tax Type, Tax Level, Tax Rate, Calculation Type.

Fixed Length *outputfilename.csf* file for customer billing.

Sorted and condensed by Customer Number, Tax Type, Tax Level, Tax Rate

FILE FORMAT KEY

Figure 5-17 srtcomcust20 SSF File Format Key	
Description	Type
Country	Alpha
State	Alpha
County	Alpha
Locality	Alpha
PCode	Numeric
Tax Type	Numeric
Tax Level	Numeric
Tax Rate	Numeric
Calculation Type	Alpha
Tax Amount	Numeric
Gross Sale	Numeric
Exempt	Numeric
Adjustments	Numeric
Taxable Measure	Numeric

Figure 5-17 srtcomcust20 SSF File Format Key	
Description	Type
Minutes	Numeric
Lines	Numeric
Customer Number	Alpha Numeric

Figure 5-18 srtcomcust20 CSF File Format Key			
Description	Type	Length	Positions
Customer Number	Numeric	20	1-20
Tax Type	Numeric	6	21-26
Tax Level	Numeric	1	27
Tax Amount	Numeric	12	28-39
File Record Length		39	

ARGUMENTS:

-z <pkzip executable> - path to zip executable file*
 N or n – CSF file is not sorted
 extra log file name – use instead of log.sum
 –cl creates a combined compliance log file with a .scl extension, containing both the billable and non-billable amounts instead of a separate log file for each.

* The pkzip executable option is only available with the windows platform.

NOTES:

- The Tax Rate field is sensitive to positive and negative values
- The log.sum file is NOT created
- WARNING - EZTax.log is DELETED. Make backups in case reruns are needed.

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

5.5.21 srtcomma20.exe

COMMAND LINE

Srtcomma20 [-z <pkzip executable>] *outputfilename [nN]* [*extra log file name*] [-cl]

DESCRIPTION

The srtcomma20.exe command reads the log file specified in filelocs.txt and produces a comma delimited text file *outputfilename.ssf* for tax compliance filing and a fixed length text file *outputfilename.csf* for customer billing.

The .csf file is used for import into the billing system to populate customer bills. The comma-delimited .ssf file is sorted at the jurisdictional level.

INPUT

Reads the *logfilename.log* file as defined by filelocs.txt or AFC Initialization Function.

log.sum – optional input from previous run or uses the extra log file name from the command line.

OUTPUT

log.sum – sorted and condensed log file as optional input for next run

Comma Delimited *outputfilename.ssf* file for tax compliance filing.

Sorted and condensed by PCode, Tax Type, Tax Level, Tax Rate

Fixed Length *outputfilename.csf* file for customer billing.

Sorted and condensed by Customer Number, Tax Type, Tax Level, Tax Rate

FILE FORMAT KEY

Country, State, County, Locality, Tax type, Tax level, Tax rate, Tax amount, Gross Sale, Exempt, Adjustments, Taxable measure, Minutes

Figure 5-19 srtcomma20 SSF File Format Key	
Description	Type
Country	Alpha
State	Alpha
County	Alpha
Locality	Alpha
Tax Type	Numeric
Tax Level	Numeric
Tax Rate	Numeric
Tax Amount	Numeric
Gross Sale	Numeric
Exempt	Numeric
Adjustments	Numeric
Taxable Measure	Numeric

Figure 5-19 srtcomma20 SSF File Format Key

Description	Type
Minutes	Numeric

Figure 5-20 srtcomma20 CSF File Format Key

Description	Type	Length	Positions
Customer Number	Alpha	20	1-20
Tax Type	Numeric	6	21-26
Tax Level	Numeric	1	27
Tax Amount Sign	Alpha	1	28
Tax Amount	Numeric	11.5	29-39
File Record Length		39	

ARGUMENTS:

-z <pkzip executable> - path to zip executable file*

N or n – CSF file is not sorted

extra log file name – use instead of log.sum

-cl creates a combined compliance log file with a .scl extension, containing both the billable and non-billable amounts instead of a separate log file for each.

* The pkzip executable option is only available with the windows platform.

NOTES:

- The Tax Rate field is sensitive to positive and negative values
- The log.sum file is NOT created
- WARNING - EZTax.log is DELETED. Make backups in case reruns are needed.

SAMPLE DATA

USA,,, 6, 0, 0.030000, 29881.866905, 996062.252444, 0.000000, 0.000000, 996062.252444, 14859213.000000
USA,,, 7, 0, 0.007600, 10116.132171, 1331070.000000, 0.000000, 0.000000, 1331070.000000, 20410578.000000
USA,,, 18, 0, 0.031400, 11145.492379, 354952.000000, 0.000000, 0.000000, 354952.000000, 5446827.000000
USA, AL, , , 16, 1, 0.067000, 969.972210, 14477.196802, 0.000000, 0.000000, 14477.196802, 220238.671875
USA, AZ, , , 1, 1, 0.050000, 0.806080, 16.121600, 0.000000, 0.000000, 16.121600, 245.280014
USA, AZ, , , 10, 1, 0.012500, 9.068400, 725.472000, 0.000000, 0.000000, 725.472000, 11037.642578
USA, AZ, , , 12, 1, 0.011000, 7.980192, 725.472000, 0.000000, 0.000000, 725.472000, 11037.642578
USA, AZ, APACHE, , 1, 1, 0.050000, 0.806080, 16.121600, 0.000000, 0.000000, 16.121600, 245.280014
USA, AZ, APACHE, , 1, 2, 0.027000, 0.435283, 16.121600, 0.000000, 0.000000, 16.121600, 245.280014
USA, AZ, APACHE, SPRINGERVILLE, 1, 1, 0.050000, 0.705320, 14.106400, 0.000000, 0.000000, 14.106400, 214.620010

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

5.5.22 srtcomma20l.exe

COMMAND LINE

```
srtcomma20l [-z <pkzip executable>] outputfilename <nN> <-p> <-s> <pcsf> <extralogfilename> [-cl]
```

DESCRIPTION

The srtcomma20l.exe command reads the log file specified in filelocs.txt and produces a comma delimited text file *outputfilename.ssf* for tax compliance filing and a fixed length text file *outputfilename.csf* for customer billing, separating adjustments.

The .csf file is used for import into the billing system to populate customer bills. The comma-delimited .ssf file is sorted at the jurisdictional level. The .ssf file also contains the number of lines.

INPUT

Reads the *logfilename.log* file as defined by filelocs.txt or AFC Initialization Function.
log.sum – optional input from previous run or uses the extra log file name from the command line.

OUTPUT

Fixed Length *outputfilename.ssf* file for tax compliance filing.
Sorted and condensed by PCode, Tax Type, Tax Level, Tax Rate.

Fixed Length *outputfilename.csf* file for customer billing.
Sorted and condensed by Customer Number, Tax Type, Tax Level, Tax Rate, Tax Amount.

FILE FORMAT KEY

Country, State, County, Locality, Tax type, Tax level, Tax rate, Tax amount, Gross Sale, Exempt, Adjustments, Taxable measure, Minutes, Lines

WITH -p OPTION SPECIFIED

PCode, Tax type, Tax level, Tax rate, Tax amount, Gross Sale, Exempt, Adjustments, Taxable measure, Minutes, Lines

**Figure 5-21 srtcomma20l SSF File Format Key
without -p option**

Description	Type
Country	Alpha
State	Alpha
County	Alpha
Locality	Alpha

Figure 5-21 srtcomma20I SSF File Format Key without -p option	
Description	Type
Tax Type	Numeric
Tax Level	Numeric
Tax Rate	Numeric
Tax Amount	Numeric
Gross Sale	Numeric
Exempt	Numeric
Adjustments	Numeric
Taxable Measure	Numeric
Minutes	Numeric
Lines	Numeric

Figure 5-22 srtcomma20I SSF File Format Key with -p option	
Description	Type
PCode	Numeric
Tax Type	Numeric
Tax Level	Numeric
Tax Rate	Numeric
Tax Amount	Numeric
Gross Sale	Numeric
Exempt	Numeric
Adjustments	Numeric
Taxable Measure	Numeric
Minutes	Numeric
Lines	Numeric

Figure 5-23 srtcomma20I CSF File Format Key			
Description	Type	Length	Positions
Customer Number	Alpha	20	1-20
Tax Type	Numeric	6	21-26
Tax Level	Numeric	1	27
Tax Amount Sign	Alpha	1	28
Tax Amount	Numeric	11.5	29-39
File Record Length		39	

ARGUMENTS

-z <pkzip executable> - path to zip executable file*

N or n – .csf file is unsorted

-p – add PCode to ssf file. Does not add PCode to csf file

-s – produce only the ssf. The .csf file is not created.

-pcsf – add PCode to csf file.

extra log file name – use instead of log.sum

-cl creates a combined compliance log file with a .scl extension, containing both the billable and non-billable amounts instead of a separate log file for each.

* The pkzip executable option is only available with the windows platform.

NOTES:

- The Tax Rate field is sensitive to positive and negative values
- The log.sum file is NOT created
- WARNING - EZTax.log is DELETED. Make backups in case reruns are needed.

SAMPLE DATA

```
USA,,, 6, 0, 0.030000, 29881.866905, 996062.252444, 0.000000, 0.000000, 996062.252444, 14859213.000000, 0
USA,,, 7, 0, 0.007600, 10116.132171, 1331070.000000, 0.000000, 0.000000, 1331070.000000, 20410578.000000, 0
USA,,, 18, 0, 0.031400, 11145.492379, 354952.000000, 0.000000, 0.000000, 354952.000000, 5446827.000000, 0
USA, AL, , , 16, 1, 0.067000, 969.972210, 14477.196802, 0.000000, 0.000000, 14477.196802, 220238.671875, 0
USA, AZ, , , 1, 1, 0.050000, 0.806080, 16.121600, 0.000000, 0.000000, 16.121600, 245.280014, 0
USA, AZ, , , 10, 1, 0.012500, 9.068400, 725.472000, 0.000000, 0.000000, 725.472000, 11037.642578, 0
USA, AZ, , , 12, 1, 0.011000, 7.980192, 725.472000, 0.000000, 0.000000, 725.472000, 11037.642578, 0
USA, AZ, APACHE, , 1, 1, 0.050000, 0.806080, 16.121600, 0.000000, 0.000000, 16.121600, 245.280014, 0
USA, AZ, APACHE, , 1, 2, 0.027000, 0.435283, 16.121600, 0.000000, 0.000000, 16.121600, 245.280014, 0
USA, AZ, APACHE, SPRINGERVILLE, 1, 1, 0.050000, 0.705320, 14.106400, 0.000000, 0.000000, 14.106400, 214.620010, 0
```

SAMPLE DATA WITH -p OPTION

```
0, 6, 0, 0.030000, 29881.866905, 996062.252444, 0.000000, 0.000000, 996062.252444, 14859213.000000, 0
0, 7, 0, 0.007600, 10116.132171, 1331070.000000, 0.000000, 0.000000, 1331070.000000, 20410578.000000, 0
0, 18, 0, 0.031400, 11145.492379, 354952.000000, 0.000000, 0.000000, 354952.000000, 5446827.000000, 0
100, 16, 1, 0.067000, 969.972210, 14477.196802, 0.000000, 0.000000, 14477.196802, 220238.671875, 0
125300, 1, 1, 0.050000, 0.806080, 16.121600, 0.000000, 0.000000, 16.121600, 245.280014, 0
125300, 10, 1, 0.012500, 9.068400, 725.472000, 0.000000, 0.000000, 725.472000, 11037.642578, 0
125300, 12, 1, 0.011000, 7.980192, 725.472000, 0.000000, 0.000000, 725.472000, 11037.642578, 0
125400, 1, 1, 0.050000, 0.806080, 16.121600, 0.000000, 0.000000, 16.121600, 245.280014, 0
125400, 1, 2, 0.027000, 0.435283, 16.121600, 0.000000, 0.000000, 16.121600, 245.280014, 0
125500, 1, 1, 0.050000, 0.705320, 14.106400, 0.000000, 0.000000, 14.106400, 214.620010, 0
```

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

5.5.23 srtcomma20Id.exe

COMMAND LINE

```
srtcomma20Id [-z <pkzip executable>] outputfilename <nN> <-p> <-s> <pcsf> <extralogfilename>  
[-cl] [-tc]
```

DESCRIPTION

The srtcomma20Id.exe command reads the log file specified in filelocs.txt and produces a comma delimited text file *outputfilename.ssf* for tax compliance filing and a fixed length text file *outputfilename.csf* for customer billing, separating adjustments.

The .csf file is used for import into the billing system to populate customer bills. The comma-delimited .ssf file is sorted at the jurisdictional level. The .ssf file also contains the number of lines.

Note: srtcomma20Id handles adjustments differently than srtcomma20I does by subtracting the adjustment from the Gross Sales before the Net Taxable Amount is calculated.

INPUT

Reads the *logfilename.log* file as defined by filelocs.txt or AFC Initialization Function.
log.sum – optional input from previous run or uses the extra log file name from the command line.

OUTPUT

Fixed Length *outputfilename.ssf* file for tax compliance filing.
Sorted and condensed by PCode, Tax Type, Tax Level, Tax Rate.

Fixed Length *outputfilename.csf* file for customer billing.
Sorted and condensed by Customer Number, Tax Type, Tax Level, Tax Rate, Tax Amount.

FILE FORMAT KEY

Country, State, County, Locality, Tax type, Tax level, Discount type, Calculation type, Tax rate, Tax amount, Gross Sale, Exempt, Adjustments, Taxable measure, Minutes, Lines

WITH -p OPTION SPECIFIED

PCode, Tax type, Tax level, Discount type, Calculation type, Tax rate, Tax amount, Gross Sale, Exempt, Adjustments, Taxable measure, Minutes, Lines

**Figure 5-24 srtcomma20ld SSF File Format Key
without –p option**

Description	Type
Country	Alpha
State	Alpha
County	Alpha
Locality	Alpha
Tax Type	Numeric
Tax Level	Numeric
Discount Type	Numeric
Calculation Type	Numeric
Tax Rate	Numeric
Tax Amount	Numeric
Gross Sale	Numeric
Exempt	Numeric
Adjustments	Numeric
Taxable Measure	Numeric
Minutes	Numeric
Lines	Numeric

**Figure 5-25 srtcomma20ld SSF File Format Key
with –p option**

Description	Type
PCode	Numeric
Tax Type	Numeric
Tax Level	Numeric
Discount Type	Numeric
Calculation Type	Numeric
Tax Rate	Numeric
Tax Amount	Numeric
Gross Sale	Numeric
Exempt	Numeric
Adjustments	Numeric
Taxable Measure	Numeric
Minutes	Numeric
Lines	Numeric

Figure 5-26 srtcomma20l CSF File Format Key

Description	Type	Length	Positions
Customer Number	Alpha	20	1-20
Tax Type	Numeric	6	21-26
Tax Level	Numeric	1	27
Tax Amount Sign	Alpha	1	28
Tax Amount	Numeric	11.5	29-39
Tax Category (optional)	Alpha	50	40-69
File Record Length		39 or 89	

ARGUMENTS

-z <pkzip executable> - path to zip executable file*
N or n – .csf file is unsorted
-p – add PCode to ssf file. Does not add PCode to csf file
-s – produce only the ssf. The .csf file is not created.
-pcsf – add PCode to csf file.
extra log file name – use instead of log.sum
-cl creates a combined compliance log file with a .scl extension, containing both the billable and non-billable amounts instead of a separate log file for each.
-tc adds the tax category description to the end of each line

* The pkzip executable option is only available with the windows platform.

NOTES:

- The Tax Rate field is sensitive to positive and negative values
- The log.sum file is NOT created
- WARNING - EZTax.log is DELETED. Make backups in case reruns are needed.

SAMPLE DATA

USA, NY, NEW YORK, NEW YORK, 1, 1, 0, 1, 0.042500, 50.950600, 1198.837637, 0.000000, 0.000000, 1198.837637, 480.0, 0
USA, NY, NEW YORK, NEW YORK, 1, 3, 0, 1, 0.041250, 49.452053, 1198.837637, 0.000000, 0.000000, 1198.837637, 480.0, 0
USA, PA, BERKS, KEMPTON, 1, 1, 0, 1, 0.060000, 58.370907, 972.848456, 0.000000, 0.000000, 972.848456, 240.0, 0
USA, PA, SCHUYLKILL, NEW PHILADELPHIA, 1, 1, 0, 1, 0.060000, 58.370907, 972.848456, 0.000000, 0.000000, 972.848456, 240.0, 0

SAMPLE DATA WITH -p OPTION

1040600, 1, 1, 0, 1, 0.060000, 34.397387, 573.289778, 0.000000, 0.000000, 573.289778, 240.0, 0
1054600, 1, 1, 0, 1, 0.060000, 34.397387, 573.289778, 0.000000, 0.000000, 573.289778, 240.0, 0
1615500, 1, 1, 0, 1, 0.050000, 25.328978, 506.579556, 0.000000, 0.000000, 506.579556, 480.0, 0
2395900, 1, 1, 0, 1, 0.060000, 111.182681, 1853.044679, 0.000000, 0.000000, 1853.044679, 480.0, 0
2604100, 1, 1, 0, 1, 0.042500, 50.950600, 1198.837637, 0.000000, 0.000000, 1198.837637, 480.0, 0
2604100, 1, 3, 0, 1, 0.041250, 49.452053, 1198.837637, 0.000000, 0.000000, 1198.837637, 480.0, 0
3232600, 1, 1, 0, 1, 0.060000, 58.370907, 972.848456, 0.000000, 0.000000, 972.848456, 240.0, 0
3415300, 1, 1, 0, 1, 0.060000, 58.370907, 972.848456, 0.000000, 0.000000, 972.848456, 240.0, 0

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

5.5.24 srtcommadetail.exe

COMMAND LINE

```
srtcommadetail [-z <pkzip executable>] outputfilename [nN] <-noCSF> <extralogfile> [-cl]
```

DESCRIPTION

The srtcommadetail.exe command reads the log file specified in filelocs.txt and produces a comma delimited text file *outputfilename.ssf* for tax compliance filing and a fixed length text file *outputfilename.csf* for customer billing.

INPUT

Reads the *logfilename.log* file as defined by filelocs.txt or AFC Initialization Function.
log.sum – optional input from previous run.

OUTPUT

Comma Delimited *outputfilename.ssf* file for tax compliance filing.
Sorted and condensed by PCode, Tax Type, Tax Level, Tax Rate

Fixed Length *outputfilename.csf* file for customer billing.
Sorted and condensed by Customer Number, Tax Type, Tax Level, Tax Rate

FILE FORMAT KEY

Country, State, County, Locality, PCode, Tax type, Tax Type Description, Tax level, Tax Level Description, Tax rate, Tax amount, Gross Sale, Exempt, Adjustments, Taxable measure, Minutes

Figure 5-27 srtcommadetail SSF File Format Key	
Description	Type
Country	Alpha
State	Alpha
County	Alpha
Locality	Alpha
PCode	Numeric
Tax Type	Numeric
Tax Type Description	Alpha
Tax Level	Numeric
Tax Level Description	Alpha
Tax Rate	Numeric
Tax Amount	Numeric
Gross Sale	Numeric
Exempt	Numeric
Adjustments	Numeric
Taxable Measure	Numeric
Minutes	Numeric

Figure 5-28 srtcommadetail CSF File Format Key			
Description	Type	Length	Positions
Customer Number	Alpha	20	1-20
Tax Type	Numeric	6	21-26
Tax Level	Numeric	1	27
Tax Amount Sign	Alpha	1	28
Tax Amount	Numeric	11.5	29-39
File Record Length		39	

ARGUMENTS

-z <pkzip executable> - path to zip executable file*
<output filename>
n or N – csf file is not sorted
nocsf – csf file is not created
extra log file name – use instead of log.sum
–cl creates a combined compliance log file with a .scl extension, containing both the billable and non-billable amounts instead of a separate log file for each.

* The pkzip executable option is only available with the windows platform.

NOTES:

- The Tax Rate field is sensitive to positive and negative values
- The log.sum file is NOT created
- WARNING - EZTax.log is DELETED. Make backups in case reruns are needed.

SAMPLE DATA

```

USA, , , 0, 6, Federal Excise Tax, 0, Federal, 0.030000, 63.946941, 2131.564739, 0.000000, 0.000000, 2131.564739, 2376516.0
USA, CA, , , 253500, 9, P.U.C. Fee, 1, State, 0.001100, 0.072776, 66.159998, 0.000000, 0.000000, 66.159998, 155916.0
USA, CA, , , 253500, 10, E911 Tax, 1, State, 0.006500, 0.450252, 69.269518, 0.000000, 0.000000, 69.269518, 155916.0
USA, CA, , , 253500, 19, State High Cost Fund, 1, State, 0.024300, 1.607688, 66.159998, 0.000000, 0.000000, 66.159998, 155916.0
USA, CA, , , 253500, 21, CA Teleconnect Fund, 1, State, 0.001600, 0.105856, 66.159998, 0.000000, 0.000000, 66.159998, 155916.0
USA, CA, , , 253500, 60, CA High Cost Fund A, 1, State, 0.001500, 0.099240, 66.159998, 0.000000, 0.000000, 66.159998, 155916.0
USA, CO, , , 427200, 13, State Universal Service Fund, 1, State, 0.029000, 0.026172, 0.902500, 0.000000, 0.000000, 0.902500, 1920.0
USA, CO, DENVER, , 442100, 4, District Tax, 2, County, 0.001000, 0.000211, 0.210945, 0.000000, 0.000000, 0.210945, 246.0

```

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

5.5.25 srtrev20l.exe

COMMAND LINE

```
srtrev20l [-z <pkzip executable>] outputfilename <nN> <-p> <-s> <pcsf> <extralogfilename> [-cl] [-tc]
```

DESCRIPTION

The srtrev20l.exe command reads the log file specified in filelocs.txt and produces a comma delimited text file *outputfilename.ssf* for tax compliance filing and a fixed length text file *outputfilename.csf* for customer billing, separating adjustments. It is intended for use with log files created using tax inclusive calculation functionality in AFC.

The .csf file is used for import into the billing system to populate customer bills. The comma-delimited .ssf file is sorted at the jurisdictional level. The .ssf file also contains the number of lines.

INPUT

Reads the *logfilename.log* file as defined by filelocs.txt or AFC Initialization Function.
log.sum – optional input from previous run or uses the extra log file name from the command line.

OUTPUT

Fixed Length *outputfilename.ssf* file for tax compliance filing.
Sorted and condensed by PCode, Tax Type, Tax Level, Tax Rate.

Fixed Length *outputfilename.csf* file for customer billing.
Sorted and condensed by Customer Number, Tax Type, Tax Level, Tax Rate, Tax Amount.

FILE FORMAT KEY

Country, State, County, Locality, Tax type, Tax level, Tax rate, Tax amount, Gross Sale, Exempt, Adjustments, Taxable measure, Minutes, Lines, Sale amount

WITH -p OPTION SPECIFIED

PCode, Tax type, Tax level, Tax rate, Tax amount, Gross Sale, Exempt, Adjustments, Taxable measure, Minutes, Lines

Figure 5-29 srtrev20I SSF File Format Key without -p option	
Description	Type
Country	Alpha
State	Alpha
County	Alpha
Locality	Alpha
Tax Type	Numeric
Tax Level	Numeric
Tax Rate	Numeric
Tax Amount	Numeric
Gross Sale	Numeric
Exempt	Numeric
Adjustments	Numeric
Taxable Measure	Numeric
Minutes	Numeric
Lines	Numeric
Sale Amount	Numeric

Figure 5-30 srtrev20I SSF File Format Key with -p option	
Description	Type
PCode	Numeric
Tax Type	Numeric
Tax Level	Numeric
Tax Rate	Numeric
Tax Amount	Numeric
Gross Sale	Numeric
Exempt	Numeric
Adjustments	Numeric
Taxable Measure	Numeric
Minutes	Numeric
Lines	Numeric
Sale Amount	Numeric

Figure 5-31 srtrev20I CSF File Format Key			
Description	Type	Length	Positions
Customer Number	Alpha	20	1-20
Tax Type	Numeric	6	21-26
Tax Level	Numeric	1	27
Tax Amount Sign	Alpha	1	28
Tax Amount	Numeric	11.5	29-39
Tax Category (optional)	Alpha	50	40-69
Base Sale Amount (optional)	Numeric	8	40-47 or 70-77
File Record Length		39, 47, 69, or 77	

ARGUMENTS

-z <pkzip executable> - path to zip executable file*
 N or n – .csf file is unsorted
 -p – add PCode to ssf file. Does not add PCode to csf file
 -s – produce only the ssf. The .csf file is not created.
 -pcsf – add PCode to csf file.
 extra log file name – use instead of log.sum
 -cl creates a combined compliance log file with a .scl extension, containing both the billable and non-billable amounts instead of a separate log file for each.
 -tc adds the tax category description to the end of each line in the csf
 -b adds the base sale amount to the end of each line in the csf

* The pkzip executable option is only available with the windows platform.

NOTES:

- The Tax Rate field is sensitive to positive and negative values
- The log.sum file is NOT created
- WARNING - EZTax.log is DELETED. Make backups in case reruns are needed.

SAMPLE DATA

```

USA, , , 55, 0, 0.136000, 2.086860, 41.360000, 26.015440, 0.000000, 15.344560, 3.5, 0, 41.36
USA, , , 170, 0, 0.015000, 0.015000, 41.360000, 26.015440, 0.000000, 15.344560, 3.5, 0, 41.36
USA, PA, , , 44, 1, 1.000000, 1.000000, 0.000000, 0.000000, 0.000000, 0.000000, 3.5, 0, 41.36
USA, PA, ALLEGHENY, PITTSBURGH, 1, 1, 0.060000, 2.738097, 45.634953, 0.000000, 0.000000, 45.634953, 3.5, 0, 41.36
USA, PA, ALLEGHENY, PITTSBURGH, 1, 2, 0.010000, 0.456350, 45.634953, 0.000000, 0.000000, 45.634953, 3.5, 0, 41.36
USA, PA, ALLEGHENY, PITTSBURGH, 14, 1, 0.050000, 2.173093, 43.461860, 0.000000, 0.000000, 43.461860, 3.5, 0, 41.36
  
```

SAMPLE DATA WITH -p OPTION

```

0, 55, 0, 0.136000, 0.000000, 82.720000, 52.030880, 15.344560, 15.344560, 7.0, 0, 82.72
0, 170, 0, 0.015000, 0.000000, 82.720000, 52.030880, 15.344560, 15.344560, 7.0, 0, 82.72
3198500, 44, 1, 1.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 7.0, 0, 82.72
3206100, 1, 1, 0.060000, 0.000000, 91.269906, 0.000000, 45.634953, 45.634953, 7.0, 0, 82.72
3206100, 1, 2, 0.010000, 0.000000, 91.269906, 0.000000, 45.634953, 45.634953, 7.0, 0, 82.72
3206100, 14, 1, 0.050000, 0.000000, 86.923720, 0.000000, 43.461860, 43.461860, 7.0, 0, 82.72
  
```

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

strg.exe

COMMAND LINE

```
strg outputfilename <extralogfilename> [-cl]
```

DESCRIPTION

The strg.exe command reads the log file specified in filelocs.txt and produces a fixed length text file *outputfilename.ssf* for tax compliance filing. If the *extralogfilename* is specified, it is used for appending instead of the log.sum.

INPUT

Reads the *logfilename* file as defined by filelocs.txt or EZtax Initialization Function.

log.sum – optional input from previous run

<*n*> - optional input superseding log.sum

OUTPUT

Fixed Length *outputfilename.ssf* file for tax compliance filing.

Sorted and condensed by PCode, Tax Type, Tax Level, Tax Rate

FILE FORMAT KEY

Figure 5-32 strg SSF File Format Key			
Description	Type	Length	Positions
PCode	Numeric	9	1-9
Tax Type	Numeric	6	10-15
Tax Level	Numeric	1	16
Tax Rate	Numeric	8	17-24
Tax Amount Sign	Alpha	1	25
Tax Amount	Numeric	11.5	26-36
File Record Length		36	

ARGUMENTS

<*outputfilename*>

extralogfilename - optional input superseding log.sum

-cl creates a combined compliance log file with a .scl extension, containing both the billable and non-billable amounts instead of a separate log file for each.

NOTES:

WARNING - EZTax.log is DELETED. Make backups in case reruns are needed.

Removes log.sum (or extra log)

SAMPLE DATA

0	1	2	3
123456789012345678901234567890123456			

0000000000000006000003000+00098418707
0000000000000007000000580+00018387113
0000000000000018000003180+00081599822
000000000000031000000039+00001000752

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

5.5.26 upsize_log.exe

COMMAND LINE

upsize_log *logfilename1.log logfilename2.log*

DESCRIPTION

The upsize_log.exe command converts the short format *logfilename1.log* to a long format log *logfilename2.log*.

INPUT

First log file *logfilename1*.

OUTPUT

The output file produced is a log file with no format key as this is a log manipulation utility.

ARGUMENTS

None.

NOTES:

log.sum is NOT created.
EZTax.log is NOT deleted.

Refer to Transactions and Tax Types for descriptions of the data in the generated file.

6. BATCH FILE PROCESSING

AFC allows company transactions to be taxed without directly interfacing AFC with the billing system using the batch_sau command line feature. Although system performance is excellent using the batch file processing method, it is less efficient than a direct interface in the following ways:

1. The billing system must provide an additional function that creates the AFC batch file.
2. AFC must read the batch-input file. With a direct interface, this is not necessary as the billing system itself will read all required data from databases or files in order to perform rating functions.
3. The billing system must read results of the AFC process for inclusion on end customer bills and journal entries in the users accounting system.

However, this method can be the most cost effective, and provide the quickest solution to tax compliance, depending upon the billing system in use and the level of technical expertise available. This is available to both AFC SaaS Standard and AFC on-site licensed product clients.

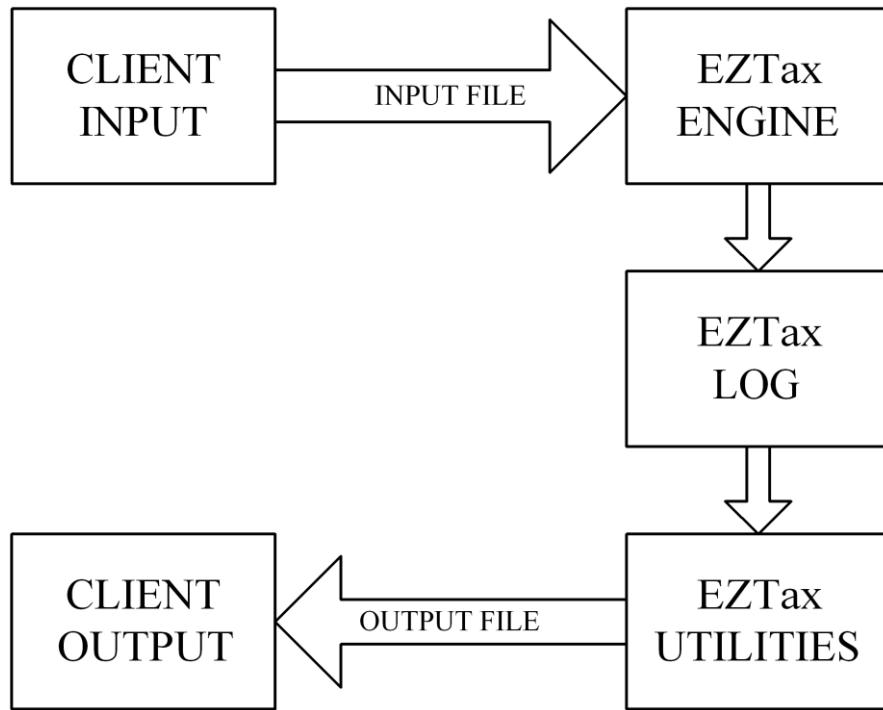
6.1 Batch Processing Description

Refer to Figure 6-1. The Batch processing method allows for transaction information in a pre-defined fixed length ASCII formatted file to be exported to and read by the AFC engine, without directly integrating to a billing system. The formatted file is submitted to the AFC Engine using the precisely defined format shown in Section 6.2. AFC processes the file and returns the taxes in a separate fixed length ASCII file generated by one of available utilities. This file can be imported by the billing system and viewed as a spreadsheet or other compatible software. The file contains key references, keyed per transaction or per account.

AFC automatically summarizes tax data generated based upon the key passed, thereby returning unique taxes per record or summarized results per customer. This simple method of interfacing is performed with relative ease and with little programming experience.

Avalara clients that perform billing on mainframe computers have also found this method beneficial because most mainframe users utilize a batch or batch-like billing process, which allows for easy insertion of a new taxation method. The taxation step is simply replaced with an export and an import process. When taxation is to be performed, all transactions are exported to a PC for taxation and the results are imported back into the mainframe.

Figure 6-1 AFC Batch Mode Model



When using the batch utility, the location of the AFC Data Base must be specified by using the filelocs.txt file. The batch utility has built in edit functions to stop any erroneous data from being submitted to the tax engine or to warn you about possible problems and still allow the transaction to be processed. An output file is generated containing warning and error messages found during the run along with the line number of the transaction.

Clients submitting CDS files in Batch fashion automatically produce an EZTax.log file. The user may select from several utilities to create a .csf file to be imported back into the billing system. Refer to Section 5.1 for aid in output utility selection.

6.2 AFC CDS Input File Specifications

The CDS fixed length input file contains one or more data records comprised of the fields detailed in the figure below. Note that a file containing this specification is provided with the monthly update.

Figure 6-2 CDF20 Client Input Data Specification							
Opt/ Req	Sec	Data Description	Type	Len	Positions	Defaults	Note
R	J	Origination	N	10	1-10		Originating NPANXX if flag=N Originating PCode if flag=P Originating State code if flag=S (numeric value)
R	J	Origination Flag	A	1	11		Set to N if using NPANXX, P if using PCode, or S if using State code
R	J	Termination	N	10	12-21		Terminating NPANXX if flag=N Terminating PCode if flag=P Terminating State code if flag=S (numeric value)
R	J	Termination Flag	A	1	22		Set to N if using NPANXX, P if using PCode, or S if using State code
R	J	Service Address	N	10	23-32		Service (BTN) NPANXX if flag=N Service PCode if flag=P Service State code if flag=S (numeric value)
R	J	Service Address Flag	A	1	33		Set to N if using NPANXX, P if using PCode, or S if using State code
R	C	Customer Type	A	1	34	B	Business, Residential Customer, Industrial, or Senior Citizen
R	T	Invoice Date	N	8	35-42		Format "CCYYMMDD"
R	T	Taxable Amount Sign	A	1	43		Format "+", "-", "p" or 'm'
R	T	Taxable Amount	N	11.5	44-54		Format "9999999999" Zero-Filled, 5 decimal places, no decimal point
R	T	Lines	N	6	55-60	0	6-character integer
R	T/S	Transaction Type	A/N	6	67-72		See Table of Transaction Types and Service Types
R	T/S	Service Type	A/N	6	73-78		See Table of Transaction Types and Service Types
R	T	Client Resale Flag	A	1	79	S	Call subject to Sale or Resale
R	C	Inc-Code	A	1	80	I	Customer is Inside or Outside an incorporated city area
R	C	Fed. Exempt	A	1	81	Blank	Call is eXempt from Federal taxes (X or blank)
R	C	State Exempt	A	1	82	Blank	Call is eXempt from State taxes (X or blank)
R	C	County Exempt	A	1	83	Blank	Call is eXempt from County taxes (X or blank)
R	C	Local Exempt	A	1	84	Blank	Call is eXempt from Local taxes (X or blank)
O	C	Customer Number (Primary Output Key)	A/N	20	85-104		Defines primary Key for Client Summary Data
R	CO	Regulated / Unregulated Flag	A	1	105	R	Reseller is Regulated or Unregulated in the state of sale
R	T	Call Duration (in minutes)	N	11.1	106-116		Format "9999999999" Zero-Filled, 1 decimal place, no decimal point

Figure 6-2 CDF20 Client Input Data Specification							
Opt/ Req	Sec	Data Description	Type	Len	Positions	Defaults	Note
R	T	Telecom Type	A	1	117	C	C - Calls, D-Debit Card Calls, P-Point of sale
R	C	Service Class Indicator	A	1	118	D	D = Primarily Long Distance, L = Primarily Local
O	C	Lifeline Flag	A	1	119	Blank	L = Lifeline customer, N = Non-Lifeline
R	CO	F _f acilities B _B ased F _F lag	A	1	120	N	F = Facilities based, N = Non-Facilities based
R	CO	F _f ranchise F _f lag	A	1	121	N	F = Franchise based, N = Non-Franchise based
R	C	B _B usiness e _e Class I _I ndicator	A	1	122	C	C = CLEC, I = ILEC

J=Taxing Jurisdiction Info. C=Customer Info. CO=Company Info. T=Transaction Info.

T/S=Transaction/Service type M= Misc.

R=Required, O=Optional

Each field within a record contains information specific to one of four categories; Taxing Jurisdiction Identification, Company, Customer and Transaction (see the table below). These four categories are addressed in detail in the following subsections.

Table 6-1 CDF Categories

Taxing Jurisdiction Identification Information	Customer Transaction Information		
	Customer Information	Company Information	Transaction Information
Origination	Customer Type	Facilities Based Flag	Invoice Date
Origination Flag	Inc-Code	Franchise Flag	Taxable Amount Sign
Termination	Fed Exempt	Regulated / Unregulated Flag	Taxable Amount
Termination Flag	State Exempt		Lines
Service Address	County Exempt		Client Resale Flag
Service Address Flag	Local Exempt		Call Duration
	Customer Number (Primary Output Key)		Telecom Type
	Service Class Indicator		Transaction Type
	Lifeline Flag		Service Type
	Business Class Indicator		

6.2.1 Taxing Jurisdiction Identification Information

The Origination, Origination Flag, Termination, Termination Flag, Service Address (also referred to as the “bill-to” address) and Service Address Flag fields of the record are discussed in this section.

The batch interface method is designed to allow for the transaction jurisdiction service address, origination point and termination point to be entered as NPANXXs, PCodes or State Code in the first fields of the transaction record. (Note that there are advantages to using the PCode as described in Section 3.4.1).

NPANXXs are the first six-digits of a phone number in North America (area code and prefix). They are administered under the North American Numbering Plan and associated with locations thereby making them useful for jurisdiction identification. However, it is important to be aware that many NPANXXs serve jurisdictions foreign to the jurisdiction they are associated with.

PCodes are proprietary permanent jurisdiction codes created by Avalara that allow AFC software users to populate their databases with proper jurisdiction assignments. When jurisdiction codes change, they are re-mapped by Avalara and the PCode is updated, providing clients with up to date information without making any changes themselves. The PCode is the preferred method of assigning and maintaining jurisdiction identification, and the time invested in including PCodes into your database records will pay off with increased accuracy and reliability.

The State Code (see the cross reference table below) should be used when restricting the tax calculations to the Federal and State level. The following table provides the valid numeric State Codes for this field. Using State Codes is discouraged as the county and locality jurisdictional information is lost and will not be applied.

Table 6-2 State Code Cross Reference Table					
State	State Code	State	State Code	State	State Code
NO STATE	0	Kansas	17	North_Dakota	35
Alabama	1	Kentucky	18	Ohio	36
Alaska	2	Louisiana	19	Oklahoma	37
Arizona	3	Maine	20	Oregon	38
Arkansas	4	Maryland	21	Pennsylvania	39
California	5	Massachusetts	22	Rhode_Island	40
Colorado	6	Michigan	23	South_Carolina	41
Connecticut	7	Minnesota	24	South_Dakota	42
Delaware	8	Mississippi	25	Tennessee	43
Washington_DC	9	Missouri	26	Texas	44
District_of_Columbia	9	Montana	27	Utah	45
Florida	10	Nebraska	28	Vermont	46
Georgia	11	Nevada	29	Virginia	47
Hawaii	12	New_Hampshire	30	Washington	48
Idaho	13	New_Jersey	31	West_Virginia	49
Illinois	14	New_Mexico	32	Wisconsin	50
Indiana	15	New_York	33	Wyoming	51
Iowa	16	North_Carolina	34		

6.2.1.1 Origination

The Origination location of the transaction to be taxed. For telecom activity, this is usually specified by the number called from, also known as the “From Number” or “Origination number”). It can be filled with the NPANXX, PCode or State Code.

6.2.1.2 Origination Flag

The Origination Flag specifies the configuration of the information in the Origination field. This field should contain N if the Origination is in NPANXX format, P if the Origination is in PCode format and S if the Origination is in State format.

6.2.1.3 Termination

The Termination location of the transaction to be taxed. For telecom activity, this is usually specified by the number called, also known as the "To Number" or the "Termination number." It can be filled with the NPANXX, PCode or State Code.

6.2.1.4 Termination Flag

The Termination Flag specifies the configuration of the information in the Termination field. This field should contain N if the Termination is in NPANXX format, P if the Termination is in PCode format and S if the Termination is in State format.

6.2.1.5 Service Address

The Service Address of the transaction to be taxed. For telecom activity, this is usually specified by the number that the call was billed to (also known as the Bill To Number or BTN) or the location of service. It can be filled with the NPANXX, PCode or State Code.

6.2.1.6 Service Address Flag

The Service Address Flag specifies the configuration of the information in the Service Address field. This field should contain N if the Service Address is in NPANXX format, P if the Service Address is in PCode format and S if the Service Address is in State format.

6.2.2 Customer Information

The transaction customer information is supplied using the following fields; Customer Type, Inc-Code, Fed Exempt, State Exempt, County Exempt, Local Exempt, Service Class Indicator, Lifeline Flag, Federal Exemption JCode, State Exemption JCode, County Exemption JCode, Locality Exemption JCode, Invoice Number, Service Level Number and Business Class Indicator.

6.2.2.1 Customer Type

This field is used to specify the type of customer involved in the transaction. The Customer Type is selected from one of the following four options.

1. Business - When transactions are made at a place of business.
2. Residential - When transactions are made by a customer for home use.
3. Industrial - When transactions are made at an industrial business.
4. Senior Citizen - When transactions made by a customer meeting the jurisdiction requirements to be considered a senior citizen and qualify for senior citizen tax breaks.

6.2.2.2 Inc-Code

The Inc-Code field is used to specify whether the customer involved in this transaction is inside (specified with an "I" in the one character length field) or outside (specified with an "O" in the one character length field) of the incorporated area designated as their location. The tax may or may not be affected by this designator depending upon whether or not the unincorporated areas are taxed in the same manner as the incorporated areas.

6.2.2.3 Exemption Levels

The exemption level is the jurisdictional level of the taxing authority that defines the tax. It is used to exempt taxes at specific federal, state, county and/or local taxes.

Fed Exempt

The Federal Exempt field is used to specify a Federal level tax exemption.

State Exempt

The State Exempt field is used to specify a State level tax exemption.

County Exempt

The County Exempt field is used to specify a County level tax exemption.

Local Exempt

The Local Exempt field is used to specify a Local level tax exemption.

6.2.2.4 Customer Number (Primary Output Key)

The Primary Output Key (POK) is a 20-character text field that is not manipulated during processing and stored as part of the log file record.

It can be used as part of the sorting key (see Figure 6-3) when using some utilities, allowing for the combining of records based upon this and other fields. It is useful when it is desired to have like taxes from different transactions combined, to have taxes from each transaction detailed individually in a report or to have each transaction detailed at the customer level.

Figure 6-3 Primary Output Key

Uniquekey.cds

0	1248900P2009108550FRS	34	106 020100815+	100000000	11	BillSoft, Inc. - 1	
0	1248900P	1248900PRS	34	106 020100815-	100000000	11	BillSoft, Inc. - 2
0	1248900P	1248900PRS	34	106 020100815p	100000000	11	BillSoft, Inc. - 3
0	1248900P	1248900PRS	34	106 020100815m	100000000	11	BillSoft, Inc. - 4

Uniquekey.csv

BillSoft, Inc.-1	0000011+00000166465
BillSoft, Inc.-1	0000012+00000034549
BillSoft, Inc.-1	0000013+00000035335
BillSoft, Inc.-1	0000060+00000094226
BillSoft, Inc.-1	0000102+00000059900
BillSoft, Inc.-1	0000131+00000145856
BillSoft, Inc.-2	0000060+00000000000
BillSoft, Inc.-3	0000060+00000000000
BillSoft, Inc.-4	0000060+00000000000
BillSoft, Inc.-5	0000011+00000038136
BillSoft, Inc.-5	0000012+00000007915
BillSoft, Inc.-5	0000013+00000008095
BillSoft, Inc.-5	0000060+00000021586
BillSoft, Inc.-5	0000102+00000014391
BillSoft, Inc.-5	0000180+00000069550
BillSoft, Inc.-6	0000011+00000011116
BillSoft, Inc.-6	0000012+00000002307
BillSoft, Inc.-6	0000013+0000002360
BillSoft, Inc.-6	0000060+0000006292
BillSoft, Inc.-6	0000102+0000004000
BillSoft, Inc.-6	0000131+0000009740
BillSoft, Inc.-7	0000060+0000009438
BillSoft, Inc.-7	0000131+00000014610

Primary Output Key

Combined taxes

If you would want like taxes combined at the customer level, then you specify a like POK for the records you want combined.

Samekey.cdf

0	1248900P2009108550FRS	34	106 020100815+	100000000	11	BillSoft, Inc.	
0	1248900P	1248900PRS	34	106 020100815-	100000000	11	BillSoft, Inc.
0	1248900P	1248900PRS	34	106 020100815p	100000000	11	BillSoft, Inc.
0	1248900P	1248900PRS	34	106 020100815m	100000000	11	BillSoft, Inc.

Samekey.csv

BillSoft, Inc.	0000011+00000215718
BillSoft, Inc.	0000012+00000044772
BillSoft, Inc.	0000013+00000045789
BillSoft, Inc.	0000060+00000131543
BillSoft, Inc.	0000102+00000078291
BillSoft, Inc.	0000131+00000170206
BillSoft, Inc.	0000180+00000069550

Primary Output Key

6.2.2.5 Service Class Indicator

The Service Class Indicator is provided to delineate the Primary activity of an organization as either Long Distance or Local Service.

- Primary Long Distance providers are carriers vending their services with over 50% of the gross business activities in Long Distance revenue.

- Primary Local Service providers are carriers vending their services with over 50% of the gross business activities in Local Service revenue.

NOTE:

This has no effect on non-Telecom Transactions.

6.2.2.6 Lifeline Flag

The Lifeline Flag is used to indicate if a customer is a Lifeline participant.

6.2.2.7 Business Class Indicator

The Business Class Indicator field is used to specify if the business making the transaction is an Incumbent Local Exchange Company (ILEC) or a Competitive Local Exchange Company (CLEC).

- An ILEC company is engaged in selling services over company owned lines and equipment,
- A CLEC company is engaged in selling services competing with an incumbent provider.

6.2.3 Company Information

The transaction information is contained in fields found in the transaction record. The Company Identifier, Facilities-Based Flag, Franchise Flag and Regulated/Unregulated Flag are discussed in this section.

6.2.3.1 Facilities Based Flag

The Facilities Based flag specifies whether the transaction is sold over tangible facilities controlled by the seller. If the seller delivering the service owns or controls the facilities used to provide the service, then the seller is facilities based. If the seller does not own or control the facilities, the seller is non-facilities based. In some jurisdictions, tax outcomes will vary depending on whether the service is delivered over infrastructure controlled by the seller.

6.2.3.2 Franchise Flag

The Franchise flag indicates that the company provides services sold pursuant to a franchise agreement between the carrier and jurisdiction.

6.2.3.3 Regulated / Unregulated Flag

The Regulated / Unregulated flag is used to specify if the Telecommunication company and its services are regulated by the Public Utility Commission, the Federal Communication Commission and/or other government authorities.

6.2.4 Transaction Information

The transaction information is contained in fields found in the transaction record. The Invoice Date, Taxable Amount Sign, Taxable Amount, Lines, Client Resale Flag, Call Duration and Telecom Type fields of the record are discussed in this section.

6.2.4.1 Invoice Date

The Invoice Date is normally populated with the bill date. Generally accepted accounting principles dictate that liabilities should be recorded when revenues are recorded. In most cases, neither of these is recorded (or even known) until billing occurs.

However, companies with a high call volume that record revenue daily as it occurs should record the tax on the same basis (i.e. the call date should be used).

AFC compares this date to the effective date of each tax that applies to the transaction.

- If the date passed to AFC is "equal to" or "greater than" the effective tax date, the current tax rate is used.
- If the date passed to AFC is prior to the effective date, the tax rate in effect for the tax is used to generate the tax amount applicable to the transaction.
- If a transaction is passed to AFC without a date (that is, the date is set to zero), AFC will set the date to the current date.

Note: The invoice date passed to the server in the transaction by default is preserved as is. It is recommended that clients not use time zone modifiers on the invoice date.

6.2.4.2 Taxable Amount Sign

The Taxable Amount Sign field is filled with either a "+" (plus) sign to indicate a charge or a "-" (minus) sign to indicate a refund or credit. In addition, the letters "p" (plus) and "m" (minus) are respectively used to indicate the appropriate refund or credit on tax inclusive calculations.

6.2.4.3 Taxable Amount

The Taxable Amount field specifies the amount of the transaction to be taxed. This amount will be passed through AFC to rate the tax based on the specified transaction/service pair. In addition, when performing a

tax inclusive calculation, the taxable or charge amount entered by the user must reflect the desired total charge plus the total tax returned (i.e. revenue plus total tax).

6.2.4.4 Lines

When local service is provided, a transaction should be generated with the Lines field populated with the number of lines the customer subscribes to. AFC uses this information for generation of per line taxes usually associated with local E911 charges and local telecommunications relay service taxes and other assorted taxes.

6.2.4.5 Transaction Type

Refer to Section 2.6 Mapping Transactions for details of this field.

6.2.4.6 Service Type

Refer to Section 2.6 Mapping Transactions for details of this field.

6.2.4.7 Client Resale Flag

Telecommunication companies are taxed on transactions made by their clients, which in some cases can be passed on or “resold” to their customers in part or in total. The Client Resale (wholesale) flag is used to indicate whether the product or service transaction is final or if it is to be resold.

To have exempt taxes available for reporting, exemption type 3 (Sales For Resale) should be used in combination with Resale.

6.2.4.8 Call Duration

The Call Duration Field specifies the length of phone call in minutes, with one tenth of a minute precision capability. AFC uses this field for generation of taxes that are specified as per minute flat fees in some taxing jurisdictions and stores the value in the AFC log database.

6.2.4.9 Telecom Type

The Telecom Type field specifies the type of telecom transaction. Enter D for Debit calls, P for Point of Sale or C for a normal call.

It is possible to see both the sale transaction and service usage with one entry in the batch system by using the “P” Telecom Type flag. In this case, the system will put the same dollar figure through the AFC engine using the transaction/service pair 10/32 followed by the 01/01 T/S pair. Since it is unlikely that the customer will use the entire prepaid service on a single long-distance call, this is best used for an approximation on what the telecom taxes will be.

6.3 Federal or State Exclusion

The Federal or State Exclusion option allows the EZTax_20 user to prevent federal and/or state (or province) and all lower jurisdictions including county, and local taxes from being created for Call Detail Records (CDRs) from specified countries or states. If excluding a state, the Federal taxes will be calculated

from CDR records that have been designated for state exclusion. If excluding a country, no taxes, including federal taxes, will be calculated.

When running AFC in batch fashion it checks for the exclude.txt file. If the file exists, the state and country jurisdictions listed in the file will be excluded. For state exclusions, state, county and local taxes are excluded. For country (federal) exclusions, country, state, county and local taxes are excluded.

If the file does not exist, the plain ASCII text file “exclude.txt” federal/state exclusions file can be created. Please reference **Section 1.5 Exclusion Configuration File** in the **AFC Manager User Manual** for additional details and instructions on creating exclusion files.

6.4 Accumulating the Log

For AFC users that run using the batch processing to do billing daily, the EZTax.Log needs to be accumulating to run the sorting utility at the end of the month to create a tax compliance file.

Daily:

1. Run the taxation on *.CDF file.
2. Archive the EZTax.Log file to another location on the system.
3. Run the sorting utility executable. *.CSF is used for current client billing. The optional *.SSF file is intended for end of the month tax compliance filing and can be ignored if generated.
4. The next taxation run of the *.CDF file will produce a new EZTax.Log file when using some utilities since it has been emptied by the previous sorting utility.

Monthly:

Avalara supplies the EZTaxappend and extaxappendf executable utilities that can be used to combine logs into a single monthly log file. At the end of the month, select the utility that best fits your requirements and combine the appropriate achieved log files. This will produce the tax compliance file .ssf file. The optional .csf file is intended for use by billing clients and can be ignored if generated.

6.5 EZTax_20 Utility Specification

COMMAND LINE

```
EZTax_20 inputfilename.cdf [-d] [-vrpt [reportname.csv]] [-?]
```

DESCRIPTION

The EZTax_20 command is a processing utility designed to accept records submitted in batch fashion. Records are supplied as an ASCII formatted file *inputfilename.cdf*.

INPUT

Reads the *inputfilename.cdf* ASCII text file

OUTPUT

EZTax.Log - as defined by filelocs.txt.

read_err.st - any errors found while editing input file are reported here.

billing.log - date/time stamp log (appended to).

bureau_log - run totals (appended to).

EZTax.sta – status from AFC engine

inputfilename.rpt - break down by totals (not for compliance filing).

reportname.csv – validation report, by default named *inputfilename.csv*

FILE FORMAT KEY

Refer to Figure 6-2.

ARGUMENTS (in any order)

input filename : The cdf file with input records

-d : Indicates to use default transaction/service types

-vrpt [*reportfile*] : Create a comma delimited validation report. This report will show each cdf line input with what API call was made along with any remarks or errors. A sample is included in Figure 6-x.

-? : Print help screen and exit

NOTES:

The transactionservice.exp file must be in the AFC working directory.

EZTax.log is not deleted.

Filelocs.txt MUST be in the working directory.

6.6 Deprecated CDF20 File Structure

Figure 6-4 CDF20 File Structure for Deprecated Input File of Version 8

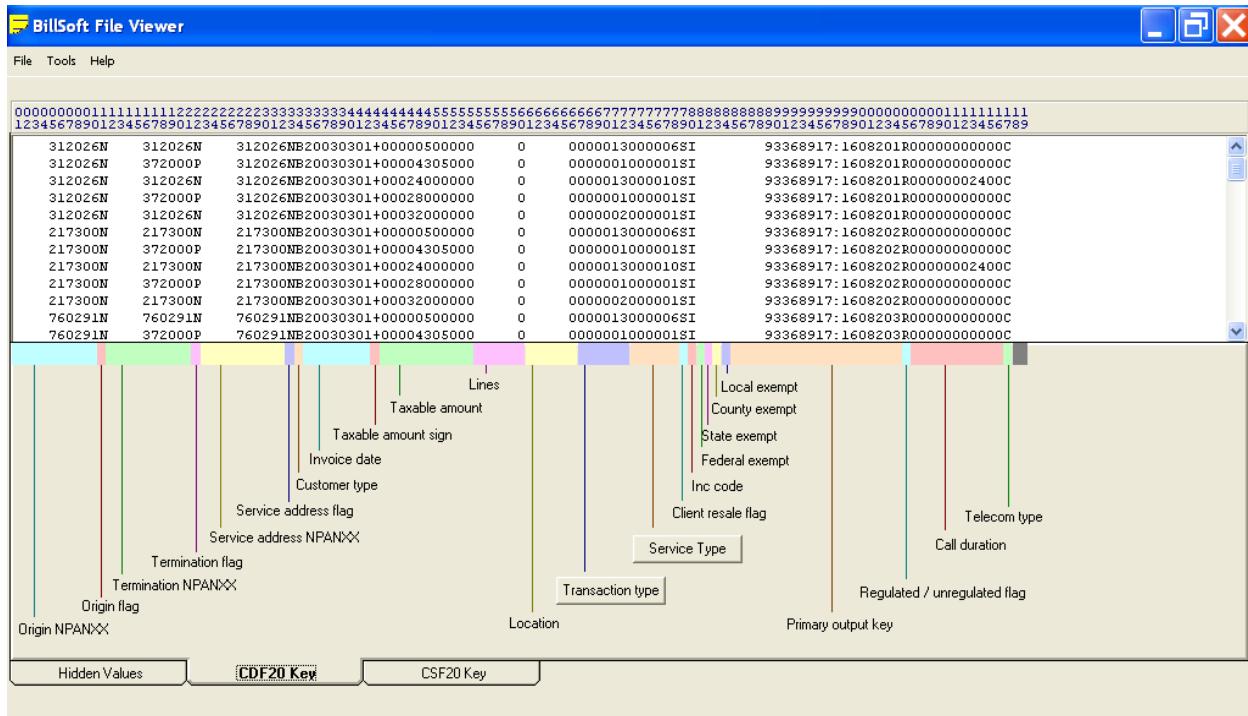


Figure 6-5 Data Specification for Deprecated Input File of Version 8

Data Description	Type	Len	In Client CDRFile	Positions	Defaults	Note
Origin	N	10	Yes	1-10	NPANXX if flag=N P code if flag=P State code if flag=S	NPANXX (Area Code+ Exchange)
Origin Flag	A	1		11		N-origin is NPANXX P-origin is npermkey S-origin state abbreviation
Termination	N	10	Yes	12-21	NPANXX if flag=N P code if flag=P State code if flag=S	Area Code+ Exchange (N) or npermkey (P) or state abNumber
Termination Flag	A	1		22		N-termination is NPANXX P-termination is npermkey S-origin state abbreviation
Service Address	N	10	Yes	23-32	NPANXX if flag=N P code if flag=P State code if flag=S	Customer NPANXX
Service Address Flag	A	1		33		N-origin is NPANXX P-origin is npermkey S-origin state abbreviation
Customer Type	A	1	Yes	34	B	Business or Residential Customer
Invoice Date	N	8	Yes	35-42		Format "CCYYMMDD"
Taxable Amount Sign	A	1	Yes	43		Format "+", "-", "p" or "m"
Taxable Amount	N	11.5	Yes	44-54		Format "9999999999" Zero-Filled, 5 decimal places, no decimal point
Lines	N	6	Yes	55-60	0	6-character integer
Transaction Type	A/N	6	Yes	67-72		See Table of Transaction Types and Service Types
Service Type	A/N	6	Yes	73-78		See Table of Transaction Types and Service Types
Client Resale Flag	A	1	Yes	79	S	Call subject to sale or resale
Inc-Code	A	1	Yes	80	I	Customer is Inside or Outside and Incorporated Locality Area
Fed. Exempt	A	1	Yes	81	Blank	Call is eXempt from Federal taxes
State Exempt	A	1	Yes	82	Blank	Call is eXempt from State taxes
County Exempt	A	1	Yes	83	Blank	Call is eXempt from County taxes
Local Exempt	A	1	Yes	84	Blank	Call is eXempt from Local taxes
Primary Output Key	A/N	20	Yes	85-104		Defines primary Key for Client Summary Data
Regulated / Unregulated Flag	A	1		105	R	Reseller is Regulated or Unregulated in the state of sale
Call Duration	N	11		106-116		Format "9999999999" Zero-Filled, 1decimal places, no decimal point
Telecom Type	A	1		117	C	C - Calls, D-Debit Calls, P - Point of Sale
File Record Length*				117		
Do not append an end-of-file character (ASCII 26) to the end of the Client Input Data File.						

* Each record Must terminate with a CR (ASCII 13) and LF (ASCII 10) located at positions 118 and 119 respectively.

7. The C/C++ API

The software for AFC is written in C/C++ and is delivered as an executable, creating a flexible programming environment. The AFC engine operates independently from the client's billing system allowing it to be easily integrated using APIs. Making use of the AFC API functions is the most efficient interface to the AFC Engine and there is very little performance difference when running the AFC taxation generation.

To integrate with AFC, programmers simply link with it and interface through a standard and well documented set of API calls. Programmers can typically perform the integration of AFC with a billing system by utilizing the documentation. If questions should arise, Avalara provides unlimited phone support for the integration of AFC and provides consulting services if desired.

Many AFC clients have accomplished integration in a matter of days and indicated that the effort with their previous system took from three months to a year. Many companies spend multiples of the yearly license cost to integrate with competing products, enduring long periods of inoperability during the integration process. The product is delivered as a DLL, shared library or shared executable allowing upgrades and enhancements of AFC to be accomplished by simply replacing a file.

Furthermore, the AFC API provides a rich and complete set of interfaces for taxation. In addition to taxation APIs, it provides the ability to do adjustments, refunds, debits, prepaids, overrides, jurisdiction information conversions and database interfaces. The APIs allow taxation using jurisdictional information provided by jurisdiction codes, NPANXX, address and zip codes.

AFC API calls are descriptively named for easy identification and selection. For instance, functions that end with "JCode" or "J" are Jurisdiction Code functions. Similar naming standards are maintained for the "PCode," "NPAN" and "Zip" functions.

7.1 Language Interfaces for AFC

AFC is written in C/C++ code. Interfaces to other software languages are provided that add another DLL for the translation from C/C++ to that other language.

AFC supports interfaces to Java, the Microsoft .NET family, RPG, COBOL and Microsoft VB6.

7.2 Configuration

Avalara offers all programs and file updates through a client login site, allowing for the download of a zip file containing the installation or update files. After an order is processed, the download zip file will be placed in the company's user folder. Check this folder regularly for new downloads, updates and announcements about Avalara's products and services.

The install wizard suggests a default file folder organization (refer to the AFC Installation Manual). Avalara encourages the use of the default settings to simplify the monthly update procedure. Changing the default settings will require a modification of each monthly update in order for it to adhere to the modified file folder organization, increasing the possibility of errors and poor performance.

The include files (header files) directory and the lib file must be incorporated into the C/C++ compiler and linker and the appropriate include files must be added to your source code to access the Avalara constants and structures. These files are in the same folder as the DLL.

7.2.1 Filelocs.txt File

AFC installs the Filelocs.upd file. This file must be moved to your working directory and renamed to filelocs.txt. Windows Explorer should be set to show file extensions so that there is no confusion between the two files. The Filelocs (File Locations) file (see Table 7-1) contains the paths to the files used by AFC.

Table 7-1 Filelocs.txt Example

PATH	DESCRIPTION
C:\BillSoft\EZTax\Data\EZTax.dat;	INPUT: EZtax Data file
C:\BillSoft\EZTax\Data\EZTax.idx;	INPUT: EZtax IDX file
C:\BillSoft\EZTax\Data\EZTax.dll;	INPUT: EZtax DLL file
EZTax.log;	I/O: EZtax log file
C:\BillSoft\EZTax\Data\EZTax.npa;	INPUT: EZtax NPANXX cross reference file
EZTax.sta;	OUTPUT: EZtax status file
tmp77777.dat;	I/O: EZtax temporary file
C:\BillSoft\EZTax\Data\EZDesc.dat;	INPUT: EZtax location description file
C:\BillSoft\EZTax\Data\ZZZIP.dat;	INPUT: EZtax ZIP Code file
C:\BillSoft\EZTax\cust_key;	I/O: EZtax customer key file
C:\BillSoft\EZTax\Data\EZTax.pcd;	
C:\BillSoft\EZTax\Data\EZTax.jtp;	

7.2.2 Filelocs.sta File

The Filelocs.sta file will be created by EZTaxInitEx when the file paths are not provided or incomplete, and if EZTaxInitEx cannot find Filelocs.txt in the current working directory. The directory where Filelocs.sta is created is the working directory that AFC is using, and this is where Filelocs.txt should be placed. Copy the Filelocs.upd file that is distributed with the AFC monthly update, rename it to Filelocs.txt, edit the locations to match your configuration, and rerun your application.

AFC produces two .sta files:

- filelocs.sta
- EZTax.sta

In addition, the utilities can produce:

- log_file.sta
- read_err.sta

All the files except EZTax.sta are normally zero length. Any other size indicates a problem. Refer to Table 7-2 for an example of the output in EZTax.sta.

Table 7-2 EZTax.sta Example

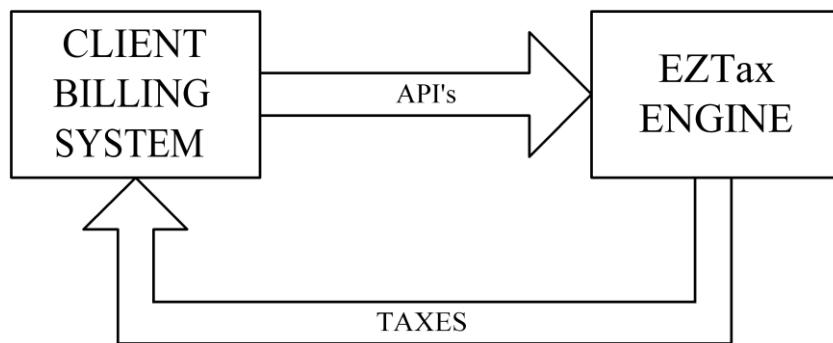
```
4/17/07 14:21:38.806 <Info>: EZtax Version Number: 9.0.0.5, platform = Windows!
04/17/07 14:21:38.806 <Warning>: Configuration line >notranslog=off< not recognized.
04/17/07 14:21:39.572 <Info>: Tax override shown below inserted successfully
04/17/07 14:21:39.572 <Info>: 0, 87, 0, 0
04/17/07 14:21:39.572 <Info>: 20060801, 1, 1
04/17/07 14:21:39.572 <Info>: 5.500000, 2147483647.000000
04/17/07 14:21:39.572 <Info>: 19000101, 1, 1
04/17/07 14:21:39.572 <Info>: 4.500000, 2147483647.000000
04/17/07 14:22:09.307 <Info>: PCODE file closed, 0 PCODE records not found
04/17/07 14:22:09.307 <Info>: JCODE file closed, 0 JCODE records not found
04/17/07 14:22:09.307 <Info>: Closing AFC Session 334260
```

7.3 General Overview of AFC API Integration

When integrated using the C/C++ APIs, the AFC Engine operates independent of the client's billing system giving the client more options. The engine is designed to allow passing of data through the program rather than integration with the client's billing system.

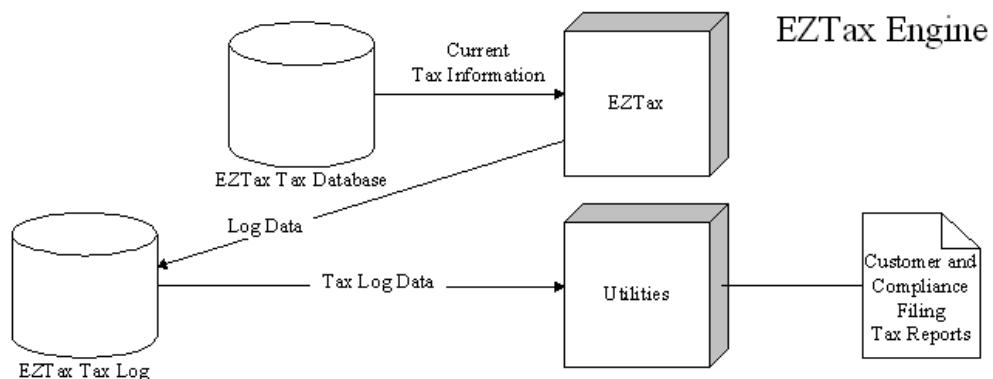
Refer the figure below. With the Application Programmer Interface (API), the billing system passes transaction information to the AFC Engine as it is being rated for billing. AFC accepts the transaction information, calculates all of the required taxes and returns the generated tax information to the billing system.

Figure 7-1 AFC API Model



Refer the figure below. In addition to calculating and generating tax information, the AFC Engine can store all of the generated tax data in AFC database log files. Utilities are provided for performing data manipulation and generating reports that facilitate tax filing and provide insight to the rating, billing and taxing processes.

Figure 7-2 AFC Engine



7.4 Detailed Discussion of AFC API Integration

Avalara provides a multi-threaded dynamic link library that is utilized to interface with AFC. Integration of an API interface opens the full capabilities of AFC.

Refer to Figure 7-3. The AFC Engine primarily accepts transaction information from the customer billing system and returns the taxes associated with the transaction. However, it can also be used to perform the following functions:

1. Perform tax adjustments.
2. Supply Jurisdiction information and JCodes based on address, Zip Code or NPANXX information.
3. Supply Jurisdiction information, address information, and convert JCodes to PCodes and PCodes to JCodes.
4. Return Tax information based on Transactions.

7.4.1 Tax Adjustments

Adjustment information is returned to the billing system and is utilized to update tax data for report generation and compliance filing. AFC also provides facilities that allow users to insert tax overrides and exempt a transaction from taxes at federal, state, county and local authority levels. In addition, AFC also provides the capability to limit exemptions to a specific jurisdiction and to exempt specific taxes.

7.4.2 Obtaining Jurisdiction and Address Information From JCodes and PCodes

Another option allows AFC to determine the taxing jurisdiction. After adding PCodes to the customer record, use the NPANXX to PCode conversion function to obtain PCodes for the NPANXXs associated with the TO and FROM numbers. AFC can use these PCodes, along with the PCode representing the “Bill To Number,” to perform tax calculations by passing the PCodes to AFC using the EZTaxPCodeEx function. In most cases it is acceptable and proper to use the PCode for a customer’s address as the “Bill To Number”. This can be helpful in cases where the BTN (Bill To Number) is not available in the CDR (Call Detail Record) source.

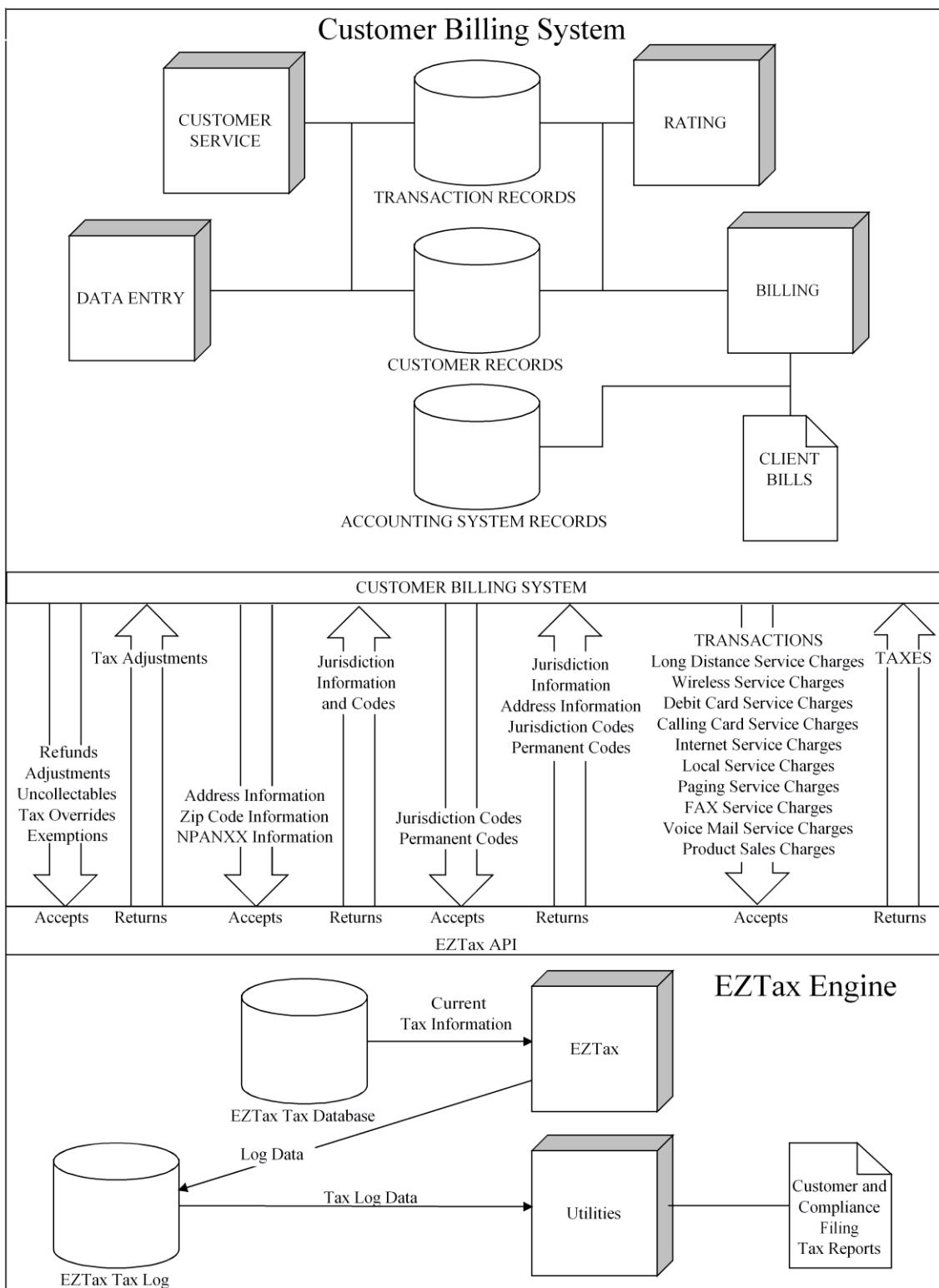
7.4.3 Obtaining Jurisdiction and Address Information From JCodes and PCodes

The AFC Address Database is available to obtain the address of a specific JCode or PCode. This is normally not used during the AFC session as the billing system normally has address information required for billing. AFC uses this database when generating reports for tax filing. The Address Database interface is provided as a valuable tool to Avalara customers.

7.4.4 Obtaining Tax Information From Transactions

AFC accepts transaction information and returns taxes to the Customer Billing System. Refer to Section [Error! Reference source not found.](#) for a detailed description of this process.

Figure 7-3 AFC API Operation Diagram



7.5 Preparing the AFC Application Programmer Interface (API) Interface

This section defines the steps required to interface AFC directly with a billing system. With this interface, the billing system passes transactions to AFC as they are being rated or billed. AFC calculates all required taxes and returns the tax information to the billing system per transaction. In addition, AFC stores all tax data generated in its databases and provides reports that facilitate tax filing and provide insight to the telecommunications rating, billing and taxing processes.

AFC functions are included in the dynamic link library EZTax2.lib. To access AFC functions, this library must be accessible when building the executable for the billing system at either link or executable build time. When using the DLL, EZTax2.dll must be in a location that will allow the system to load the DLL.

To use the Avalara's AFC Engine in a C or C++ code environment, the include file directory and the lib file must be incorporated into the client C/C++ compiler and linker.

1. The include file directory is in the same folder as the dll found at C:\BillSoft\EZTax\dll if you installed using the default directory. Add this folder to the include folders for your C++ compiler.
2. Add the following include files located in the dll folder to your source code in order to access the Avalara constants and structures.
 - a. EZTaxStruct.h - Provides data structures required to interface with AFC. Refer to Appendix A for the detailed structure.
 - b. EZTaxDefine.h, EZTaxTransType, EZTaxServType, EZTaxTaxType.h, EZTaxServType.h, EZTaxCountryType.h, EZTaxStateType.h - Provides data constants required to interface with EZtax. Refer to Appendix B for the detailed structure.
 - c. EZTaxProto.h - Contains function prototypes. Refer to Appendix C for the detailed structure.
3. Add the selected include files and the lib file to your project file list if you wish to use editing tools such as Microsoft Visual Studio to perform code completion based on the structure members.

NOTE:

If you include the EZTaxProto.h file in your source code it will include the EZTaxStruct.h and the EZTaxDefine.h.

7.6 AFC API Function Calls

An AFC session is started with a call to one of the initialization routines (EZTaxInitEx or EZTaxInitExMT, usually) to initialize the task. APIs are called to perform specified AFC functions that achieve the desired results of the session.

Once all transactions are completed, EZTaxExitSessionEx must be called for each session started to flush database buffers, de-allocate memory and perform other clean-up duties which free up resources utilized

by AFC. Sessions must be completed by a call to EZTaxExitSessionEx at least once per tax filing period to insure that the tax log is properly configured and complete before utilities that generate filing information are started.

7.6.1 Retrieving Taxes

7.6.1.1 The AFC Tax Table

The AFC table is allocated during AFC session initialization. The size of this table is dependent upon the maximum taxes that can be generated for a single transaction. As such, the size of this table can change from month to month as new taxes are generated or removed from taxing jurisdictions. It is always safe to access from 0 (zero) to [tax_count_returned -1] locations of the table. The user is cautioned to treat this as a read only area. Attempting to access locations that do not exist will result in access violations on most operating systems.

Each function that performs tax calculations or adjustments returns a count of the taxes generated for the transaction and stores the taxes in the Tax Table. The taxes can be retrieved for storage in a billing system by using the taxes_tbl struct pointer, which is activated when EZTaxInitEx is called. The pointer to the AFC table can be used to select a tax from the array of taxes_tbl objects using the following code.

```
{  
    int index;  
    double tax;  
    tax = EZTax_table[index].tax_amount;  
}
```

7.6.1.2 Sample Code to Retrieve Taxes

To retrieve taxes generated, the following code segment (shown without proper status checking) can be used with the DLL:

```
#include <BillSoft/EZTaxDefine.h>
#include <BillSoft/EZTaxProto.h>
#include <BillSoft/EZTaxStruct.h>

main()
{
    short int                  i;
    short int                  tax_count;
    int                        err_code;
    double                     tax;
    struct enhanced_taxes_tbl *EZTax_table;
    struct sau_J_Code          transaction;
    EZTaxSession                session;

    EZTax_table = EZTaxInitEx(FALSE, NULL, &session);

    // initialize the transaction here

    tax_count = EZTaxSAUJCode(session, &transaction,
                               NULL, 0, &err_code);
    if (tax_count > 0)
    {
        for(i=0; i<tax_count; i++)
        {
            tax = EZTax_table[i].tax_amount;

        // Perform billing system task or store tax data

        }
    }
    EZTaxExitSessionEx(session);
}
```

7.6.2 Multi-Threading

A user can process multiple sessions in separate threads (referred to as multi-threading) with AFC. This technique increases the processing speed and efficiency of the client billing system because each thread can start an independent AFC session and call the specified AFC functions.

For example, if there is a requirement to compute a customer's taxes on calls and services, and also to offer tax quotes, then one session can be setup with tax logging turned on to calculate the customer's bills while another session is established with tax logging turned off to offer tax quotes without the results being logged.

Another example would be to allow the client's billing system to process more than one customer base on the same run, with each customer base using a separate session. In addition, the capability exists to specify that each session will have a unique tax log in order to produce separate logs at session start up.

7.6.2.1 The latest in version 9

The latest version of AFC 9 features enhanced support for applications that use AFC in a multi-threaded environment. Applications that want to take advantage of this new support should adhere to the following guidelines:

1. AFC supports concurrent processing on separate threads by assigning each thread its own EZTaxSession object. Usage of the same EZTaxSession by multiple threads is not supported and will result in unpredictable and erroneous tax calculations.
2. Each separate EZTaxSession object should have a unique AFC log file if tax logging is enabled for that session.
3. Each separate EZTaxSession object should have a unique status file.
4. When opening AFC, use the new "EZTaxInitExMT" method. Users who previously initialized AFC using EZTaxInitEx will have one additional parameter to provide - a string that contains the directory name to be used as the AFC Default directory. If this string is NULL, AFC will continue to look in the Working Directory of the application for EZTax.cfg and filelocs.txt. Otherwise, AFC will search the supplied directory name for EZTax.cfg and filelocs.txt.
5. When an AFC Session is no longer needed, be sure to call EZTaxExitSessionEx with the AFC Session object obtained from EZTaxInitExMT. Sessions that are opened with EZTaxInitExMT will not be closed when EZTaxExit is called, and failure to call EZTaxExitSessionEx will result in a memory leak.
6. Certain functions will not work if a session is created with EZTaxInitExMT. These include EZTaxGetHandle, which translates an EZTaxSession object to an integer handle. Since the corresponding handle is not created by AFC when using the new EZTaxInitExMT method, functions that take an integer handle argument cannot be used against a multi-threaded session.
7. Other functions that take no handle or session object are also not supported, including:
 - a. EZTaxPtoFips (use EZTaxPtoFipsEx)

- b. EZTaxNtoJCode (use EZTaxNtoJCodeEx)
- c. EZTaxJtoPCode (use EZTaxJtoPCodeEx).

When sessions have been initialized by EZTaxInitExMT, an attempt to use these functions will return an error code and will not work.

8. A new function has been added to AFC so that sessions created by EZTaxInitExMT will have the functionality of AFC without resorting to an integer session handle. EZTaxOldOvrJCode allows users to use the old override structure with an EZTaxSession object.

Users that start all their sessions before any separate threads are started and before any tax calculations are performed can continue to use EZTaxInitEx, or the new function EZTaxInitDirEx. EZTaxInitDirEx provides the name for the Default AFC directory, exactly like EZTaxInitExMT. Neither of these functions is thread-safe, but the subsequent processing against the EZTaxSession objects created by these functions is thread-safe.

If one or both of these non-thread-safe functions are used to initialize all EZTaxSession objects, EZTaxExit will continue to close any open EZTaxSession objects without the need for a specific call to EZTaxExitSession for each EZTaxSession object.

7.6.3 Multiple Sessions

When using the APIs, a session is started with a call to EZTaxInitExMT. Successive calls to EZTaxInitExMT will initialize additional sessions, thereby accomplishing multi-threaded operation.

AFC writes compliance information to the EZTax.log file. The compliance logging function can be enabled or disabled with each session and each session controls its own log file. The Log records are kept in the memory to improve the efficiency and are written to the disk periodically. Several techniques can be used to control when the log is written to disk, although many applications do not need any special handling of the compliance log.

Overrides may be used with multiple sessions as each session will maintain its own overrides. The override can be specified by JCode, PCode, NPANXX and Zip Code. Since each session has its own override, the user can remove the override from a session by using the call EZTaxRestoreEx.

Each Session must have a separate Log File

Each Session should have a separate temporary File - and this is required if that session applies overrides to the tax table

7.6.4 Session Management

Several functions call EZTaxGetHandle to create session handles into memory pointers. Utilizing the session handle from the memory improves operating efficiently. The session handle references AFC sessions. It is a long integer that is pulled from a stack of unused session handles setup during the EZTaxInitEx call.

EZTaxInitEx returns a void pointer type to the session memory and is defined as EZTaxSession. Each time a session is initialized through EZTaxInitEx the session pointer is placed on a dynamically allocated stack.

The File Path structure (see Appendix B Appendix B EZTaxDefine.h) provides the ability to override any file path specified in the filelocs.txt configuration file programmatically. A subset of the paths can be overridden by specifying "NULL" for paths where the default is to be maintained. An address to this structure can be specified when initializing a session with EZTaxInitEx.

7.6.4.1 General Tips to Maintain Session Efficiency

When using sessions to make calls to the database, it is important to maintain the performance level between the billing system and the AFC engine.

The following tips have been accumulated to assist users in maintaining session efficiency.

1. Some calculations, such as requests for quotes from the Sales department, might not be appropriate to log to the compliance files. Open a separate session for quotes with no logging to achieve this.
2. If calculations have special tax rates that require an override of the AFC database, open a separate session (with a separate log file and temporary file) for the override calculations.
3. If the application terminates all sessions at the end of a billing cycle, the normal session completion will save all of the compliance data to the disk.
4. The EZTaxFlushToLogEx API can be used if more control of the log file is needed. Some specialized applications call EZTaxFlushToLogEx after every call to one of the tax calculations, to ensure that the disk copy of the log is always completely up-to-date.
5. Using the EZTaxExitSessionEx to control the log will close the session and save the compliance data to the log. If all sessions are complete then EZTaxExit will perform the same function for each session open.
6. Higher efficiency levels are obtained if you have sufficient memory (approximately 10 MB per session) to "CACHE_ALL." Using the memory CACHE improves the processing speed many times over.
7. The maximum number of sessions is 500.

8. If multiple users want to perform tax calculations simultaneously (such as a Web application) then use Session Pooling (Refer to 7.6.5).

7.6.5 Session Pooling

Session pooling is a resource managing tool that allows for multiple sessions to be held in a “pool” where each can be checked out, processed and returned one at a time.

It is modeled after the office “steno pool” where instead of assigning a stenographer to a department, a department would request one from the pool. The stenographer would leave, take the dictation and return to the pool waiting for the next call. The concept was that a company might have only 5 stenographers, but 20 departments that needed stenography service. If 20 stenographers were hired, they would probably be sitting around with no work for 75% of the work day. The drawback is that sometimes all 5 will be checked out of the pool, and a department will have to wait for a stenographer to return.

Session pools work in exactly the same manner. A transaction comes in that needs to be processed, a session is grabbed from the pool, the session processes the transaction, and the session is returned to the pool. If a second transaction comes in before the first transaction has completed, it gets the second session from the pool. The AFC Web service has multiple sessions in the session pool to handle about 20 customers of AFC SaaS Pro.

The application should be structured in the following manner:

1. Create the number of sessions required.
2. Store the session handles in a memory table with a “use flag.”
3. When a tax calculation is required, search the table to find a handle that is not in use.
4. Set the use flag to on indicating the session is “in use”.
5. Perform the tax calculation.
6. Set the use flag to off, indicating the session is “not in use”.

7.6.5.1 General Tips When Using Session Pooling

The following tips have been accumulated to assist users when using Session Pooling.

1. Periodically call EZTaxFlushToLogEx on each session to update the EZTax.log for each open session.
2. If there are more than 10 concurrent users the pool of sessions can be permitted to grow dynamically for greater flexibility.
3. The use of Invoice Mode and overrides is prohibited since the session obtained for the next transaction is unknown. Invoice Mode could be used if the session was held for all of the customer's transactions.

7.7 Sample Coding

The following sample code listings have been accumulated to assist users in performing specific tasks.

7.7.1 Using EZTaxGetRates to Build an Override

The Get Rates function can be used to accomplish an override using the following steps.

1. Call EZTaxGetRates with the desired PCode for the jurisdiction

```
Call objEZTax.EZTaxGetRates(glHandle, OLATHE_PCODE, taxes)
```

2. Search the taxes Table returned in the Jurisdiction Data to find the desired Enhanced Override for your tax type and level. This example searches for the state sales tax.

```
overrides = taxes.taxesTable
For i = 0 To taxes.taxesCount - 1
    If (overrides(i).Type = SALES_TAX)
        And (overrides(i).level = STATE_LEVEL) Then
            Call printMessage("Found State Sales Tax")
        Set stateSalesTax = overrides(i)
        foundSalesTax = True
    End If
Next I
```

3. Using the Enhanced Override object found, make the appropriate changes

```
If foundSalesTax Then
    effectiveDates = stateSalesTax.dateTable
    Set currentEffectiveDate = effectiveDates(0)
    rates = currentEffectiveDate.rateTable
    Set currentRate = rates(0)
    currentRate.tax = 0.04
```

4. Apply the override to AFC

```
Call objEZTax.EZTaxOvrJCodeEx(glHandle, OLATHE_JCODE, stateSalesTax)
Call printMessage("Rate changed to 4%")
End If
```

7.8 API Listings

APIs Listed By Function	
Sorted by Function	Description
Session Management Functions	
The following functions support session activity.	
EZTaxInitEx	Initializes an AFC Session.
EZTaxInitExMT	Initializes an AFC Multi-session. Multiple calls may be made to set up completely different sessions.
EZTaxInitDirEx	Reads AFC files and initializes the AFC system. Multiple calls may be made to set up completely independent sessions.
EZTaxInitV914	Reads AFC files and initializes the AFC system. Multiple calls may be made to set up completely independent sessions. The input structure allows all optional input files to be specified through the initialization method. This method does not use filelocs.txt and will ignore exclusion, nexus and bundle file settings in EZTax.cfg.
EZTaxInitV98	EZTaxInitV98 reads AFC files and initializes the AFC system. It differs from EZTaxInitExMT in that the tax table returned includes tax category information.
EZTaxClearExclusion	Removes all exclusions that have been set
EZTaxClearTSR	Allows API functionality to delete the data created by EZTaxGetTSR.
EZTaxClose	Once processing is completed, databases must be closed using EZTaxClose.
EZTaxExitSessionEx	Exits an AFC Session.
EZTaxExit	EZTaxExit insures that all internal buffer contents are stored on disk, memory allocated by the system is deallocated and that databases are properly closed.
EZTaxSetWorkingDir	Used to define a directory as the working directory.
EZTaxSetNexus	Sets nexus on or off for the specified states.
EZTaxSetStateExclusion	Uses the country code and a state code to set an exclusion.
EZTaxSetStateNexus	Uses a state code to set the nexus.
Jurisdiction Conversion Functions	
The following functions are provided to convert between jurisdictional interfaces. This is useful when the service address is available for a call record transaction to be taxed but only NPANXX information is available for origination and termination points.	
EZTaxCountryToPCode	EZTaxCountryToPCode takes a 3 byte country ISO code and converts it to the correct PCode for that country.
EZTaxGetAddressEx	Used to obtain the address of a specific JCode. This is normally not used during the AFC Session as the billing system normally has address information required for billing. AFC uses this database when generating reports for tax filing. The Address Database interface is provided as a valuable tool to Avalara customers.
EZTaxGetJurisdiction	Retrieves the jurisdiction where the transaction is determined to be taxed.

APIs Listed By Function	
Sorted by Function	Description
EZTaxGetTaxCatV98	EZTaxGetTaxCatV98 retrieves the tax category description for a specified tax code.
EZTaxJtoPCodeEx	Returns PCode that is cross-referenced to a JCode.
EZTaxJurisdiction	Returns the first JCode at the lowest level.
EZTaxNextAddressEx	Returns the next sequential address for a specified jurisdiction when the jurisdiction or parts of it are known by more than one address. This is true for the many jurisdictions contained within the AFC system, especially large cities. The Address Database interface is provided as a valuable tool to Avalara customers.
EZTaxPtoJCodeEx	Converts a PCode into a JCode.
EZTaxZtoJCodeEx	Used to populate customer databases records with JCodes. Avalara recommends the use of PCodes over JCodes, in which case the ZtoPCodeEx function would be used. This provides users that interface with AFC a large performance boost by providing addresses. The Zip Code Database interface is provided as a valuable tool to Avalara customers.
EZTaxZtoPCodeEx	Converts a zip code and address to a PCode.
EZTaxPtoFipsEx	Converts a PCode to a Fips Code.
EZTaxNtoJCodeEx	Obtain JCodes associated with an NPANXX.
EZTaxFtoPCodeEx	Converts a Fips Code into a PCode.
EZTaxGetCountryID	Get country ID for a PCode
EZTaxGetStateID	Get state ID for a PCode
Debit Transaction Functions	
EZTaxDebitJEx	Accepts transaction data and performs tax debit calculations. BTN, origination and termination information is passed using JCodes.
EZTaxDebitNEx	Accepts transaction data and performs tax debit calculations. BTN, origination and termination information is passed using NPANXXs.
EZTaxDebitPEx	Accepts transaction data and performs tax debit calculations. BTN, origination and termination information is passed using PCodes.
Override Functions	
The following functions are provided to override an AFC value for the remainder of a session.	
EZTaxRestoreEx	Removes all overrides that have been inserted by the user into the AFC databases. This action occurs automatically on EZTaxExit.
EZTaxOldOvrJCodeEx	EZTaxOvrJCodeEx specifies an override for a specific tax. Tax Jurisdiction information is passed via jurisdiction code. It takes a J_Code and an override structure as input and inserts a tax override into the AFC db system.
EZTaxOvrJCodeEx	Specifies an override for a specific tax. Tax Jurisdiction information is passed with a jurisdiction code. It takes a JCode and an override structure as input and inserts a tax override into the AFC db system for that session.
EZTaxOvrNPANEx	Specifies an override for a specific tax. Tax jurisdiction information is passed with an NPANXX. It takes an NPANXX and an override structure as input and inserts a tax override into the AFC db system for that session.

APIs Listed By Function	
Sorted by Function	Description
EZTaxOvrPCodeEx	Specifies an override for a specific tax. Tax jurisdiction information is passed with a PCode. It takes a PCode and an override structure as input and inserts a tax override into the AFC db system for that session.
EZTaxOvrZipEx	Specifies an override for a specific tax. Tax jurisdiction information is passed via zip code and address. It takes zip code plus 4 and address information, as well as an override structure as input and inserts a tax override into the AFC db system for that session.
Invoice Mode Functions	
The following functions are provided to support Invoice Mode Operation.	
EZTaxSetInvoiceModeEx	Turns Invoice Mode on and off.
EZTaxSetInvoiceModeV98	EZTaxSetInvoiceModeV98 sets Invoice Mode on or off - allocating or freeing memory for tables. It differs from EZTaxSetInvoiceModeEx in that the tax table returned contains tax category information.
EZTaxNextCustomerEx	Indicates that all of the transactions associated with this customer are complete. The system then clears out the transaction history table in preparation of the next set of transactions.
EZTaxJCodeEx	Accepts transaction data and performs appropriate tax calculations. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using JCodes.
EZTaxTaxInclusiveJCode	Accepts transaction data and performs a tax inclusive calculation. If the AFC Session was started with the tax log open, a transaction with the newly-calculated charge will be logged and will be reflected in tax reports. BTN, origination and termination information is passed using JCodes.
EZTaxDebitTaxInclusiveJCode	Accepts transaction data and performs a tax inclusive calculation for prepaid debit transactions. If the AFC Session was started with the tax log open, a transaction with the newly-calculated charge will be logged and will be reflected in tax reports. BTN, origination and termination information is passed using JCodes.
EZTaxThisJCodeEx	Accepts transaction data and performs appropriate tax calculations. The tax jurisdiction to use for tax calculations is specified to AFC with this method. If the AFC Session was started with the tax log open, appropriate taxes are logged and will be reflected in tax reports. Jurisdiction information is passed using a JCode.
EZTaxThisPCodeEx	Can be used to specify taxing based upon address information only by passing the encoded PCode.
EZTaxPCodeEx	Accepts transaction data and performs appropriate tax calculations. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using PCodes.
EZTaxTaxInclusivePCode	Accepts transaction data and performs a tax inclusive calculation. If the AFC Session was started with the tax log open, a transaction with the newly-calculated charge will be logged and will be reflected in tax reports. BTN, origination and termination information is passed using PCodes.

APIs Listed By Function	
Sorted by Function	Description
EZTaxDebitTaxInclusivePCode	Accepts transaction data and performs a tax inclusive calculation for prepaid debit transactions. If the AFC Session was started with the tax log open, a transaction with the newly-calculated charge will be logged and will be reflected in tax reports. BTN, origination and termination information is passed using PCodes.
EZTaxNPANEx	Accepts an NPANXX as input and calculates taxes for the transaction. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using NPANXXs.
EZTaxTaxInclusiveNPAN	Accepts transaction data and performs a tax inclusive calculation. If the AFC Session was started with the tax log open, a transaction with the newly-calculated charge will be logged and will be reflected in tax reports. BTN, origination and termination information is passed using NPANXXs.
EZTaxDebitTaxInclusiveNPAN	Accepts transaction data and performs a tax inclusive calculation for prepaid debit transactions. If the AFC Session was started with the tax log open, a transaction with the newly-calculated charge will be logged and will be reflected in tax reports. BTN, origination and termination information is passed using NPANXXs.
EZTaxZipEx	Accepts transaction data and performs appropriate tax calculations. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports.
EZTaxTaxInclusiveZip	Accepts transaction data and performs a tax inclusive calculation. If the AFC Session was started with the tax log open, a transaction with the newly-calculated charge will be logged and will be reflected in tax reports. Jurisdiction information is passed using address information.
EZTaxDebitTaxInclusiveZip	Accepts transaction data and performs a tax inclusive calculation for prepaid debit transactions. If the AFC Session was started with the tax log open, a transaction with the newly-calculated charge will be logged and will be reflected in tax reports. Jurisdiction information is passed using address information.
Credit Adjustment Functions	
The following functions are provided for implementing adjustments such as refunds, changing a customer's bill or when terminating un-collectable accounts.	
These functions accept transaction data and perform appropriate tax calculations required to refund or credit taxes for U.S. jurisdictions (Debits use EZTaxAdjDebitEx APIs). If the AFC Session was started with the tax log open, appropriate taxes reductions are logged and will be reflected in tax reports.	
EZTaxAdjDebitJEx	BTN, origination and termination information is passed using JCodes.
EZTaxAdjDebitNEx	BTN, origination and termination information is passed using NPANXXs.
EZTaxAdjDebitPEx	BTN, origination and termination information is passed using PCodes.
EZTaxAdjDebitZEx	The tax jurisdiction is specified using address information.
EZTaxAdjJCodeEx	BTN, origination and termination information is passed using JCodes.
EZTaxAdjTaxInclusiveJCode	Performs a tax inclusive calculation. Any logging will be done with the newly-calculated charge. BTN, origination and termination information is passed using JCodes.
EZTaxAdjDebitTaxInclusiveJCode	Performs a tax inclusive calculation for a prepaid debit transaction. Any logging will be done with the newly-calculated charge. BTN, origination and termination information is passed using JCodes.
EZTaxAdjNPANEx	BTN, origination and termination information is passed using NPANXXs.

APIs Listed By Function	
Sorted by Function	Description
EZTaxAdjTaxInclusiveNPAN	Performs a tax inclusive calculation. Any logging will be done with the newly-calculated charge. BTN, origination and termination information is passed using NPANXXs.
EZTaxAdjDebitTaxInclusiveNPAN	Performs a tax inclusive calculation for a prepaid debit transaction. Any logging will be done with the newly-calculated charge. BTN, origination and termination information is passed using NPANXXs.
EZTaxAdjPCodeEx	BTN, origination and termination information is passed using PCodes.
EZTaxAdjTaxInclusivePCode	Performs a tax inclusive calculation. Any logging will be done with the newly-calculated charge. BTN, origination and termination information is passed using PCodes.
EZTaxAdjDebitTaxInclusivePCode	Performs a tax inclusive calculation for a prepaid debit transaction. Any logging will be done with the newly-calculated charge. BTN, origination and termination information is passed using PCodes.
EZTaxAdjProRateJCode	Accepts a J_Code struct as input and calculates tax adjustments for the transaction, then pro-rates the taxes calculated by the given percentage for partial-month rates.
EZTaxAdjProRateNPAN	Accepts an NPAN_struct as input and calculates tax adjustments for the transaction, then pro-rates the taxes calculated by the given percentage for partial-month rates.
EZTaxAdjProRatePCode	Accepts a PCode struct as input and calculates tax adjustments for the transaction, then pro-rates the taxes calculated by the given percentage for partial-month rates.
EZTaxAdjProRateThisJCode	Accepts transaction data and performs appropriate tax calculations. The tax jurisdiction to use for tax calculations is specified to AFC with this method. If the AFC Session was started with the tax log open, appropriate taxes are logged and will be reflected in tax reports. Jurisdiction information is passed using a JCode. It pro-rates the taxes calculated by the given percentage for partial-month rates.
EZTaxAdjProRateThisPCode	Accepts transaction data and performs appropriate tax calculations. The tax jurisdiction to use for tax calculations is specified to AFC with this method. If the AFC Session was started with the tax log open, appropriate taxes are logged and will be reflected in tax reports. Jurisdiction information is passed using a PCode. It pro-rates the taxes calculated by the given percentage for partial-month rates.
EZTaxAdjProRateZip	EZTaxAdjProRateZip takes a Zip Code struct as input and calculates tax adjustments for the transaction. It pro-rates the taxes calculated by the given percentage for partial-month rates.
EZTaxAdjTaxInclusiveZip	Performs a tax inclusive calculation. Any logging will be done with the newly-calculated charge. BTN, origination and termination information is passed using address information.
EZTaxAdjDebitTaxInclusiveZip	Performs a tax inclusive calculation for a prepaid debit transaction. Any logging will be done with the newly-calculated charge. BTN, origination and termination information is passed using address information.

APIs Listed By Function	
Sorted by Function	Description
EZTaxAdjTPPEx	EZTaxAdjTPPEx accepts Tangible Personal Property (TPP) transaction data and performs appropriate tax calculations required to refund or credit taxes. The user provides up to three addresses: Ship From, Ship To and Point of Acceptance. Rules will be applied against the input addresses to determine in what jurisdiction any SALES or USE taxation would be applicable. A Nexus Table may optionally be provided. If so it will be checked to determine if nexus exists in the state found to have taxing jurisdiction. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports.
EZTaxAdjZipEx	The tax jurisdiction is specified using address information.
Interstate-Intrastate Determination Functions	
The Interstate-Intrastate functions are used to determine if a call is interstate or intrastate.	
EZTaxJTypeEx	Returns transaction type for a given pair of JCodes indicating whether the call is interstate or intrastate. The first JCode is the originating and the second the terminating JCode. Will set error code to INTERSTATE_NOT_VALID_FOR_INTERNATIONAL_CALLS If the originating country and terminating country are not the same.
EZTaxNTTypeEx	Returns transaction type for a given pair of NPANXX's indicating whether the call is interstate or intrastate. The first NPANXX is the originating and the second is the terminating NPANXX.
EZTaxPTTypeEx	Returns transaction type for a given pair of PCodes indicating whether the call is interstate or intrastate. The first PCode is the originating and the second the terminating PCode. Will set error code to INTERSTATE_NOT_VALID_FOR_INTERNATIONAL_CALLS If the originating country and terminating country are not the same.
EZTaxTPPEx	Accepts up to three addresses; Ship From, Ship To, and Point-Of-Acceptance. It requires Ship From. If others have NOT NULL values, then they must be valid. Uses rules to determine interstate or intrastate to determine which taxing jurisdiction should be used. It accepts the optional Nexus Table, which if given will be checked against the jurisdiction determined above. No taxes will be calculated unless the flag is set (by the caller) for the given state. It returns tax_count, having populated the tax log structure with any taxes found.
Get Functions	
These functions are supplied to support activities requiring the retrieval of information and data.	
EZTaxGetCustomLog	Returns the JCode that has jurisdiction at the most detailed taxing level.
EZTaxGetCustomLogCount	Returns the number of records in the current custom log table.
EZTaxGetHandle	Returns the handle object given the session.
EZTaxGetLogName	Retrieves the log filename when it is specified using the filelocs.txt. Current use is in the batch executables that open the log file and must know the filename.
EZTaxGetLogV914	Returns a pointer to a log structure, which is used for customized logging. Returns a pointer to the v9114 tax log table.
EZTaxGetLogV914Count	Returns the number of records in the current custom log table.
EZTaxGetRates	Utilizes a PCode as input to produce the table of all taxes for that jurisdiction.

APIs Listed By Function	
Sorted by Function	Description
EZTaxGetSession	Returns the session object given the handle.
EZTaxGetTaxDescription	Retrieves the tax description for a specified tax code. Current use is in utilities reporting from the log, but users can also use this new feature.
EZTaxGetTSR	Allows applications API functionality to get the transaction service report
EZTaxMaxTaxCount	Returns the maximum number of taxes that one transaction can generate.
ProRate Functions	
EZTaxProRate functions accept transaction data and perform appropriate tax calculations. If the taxing jurisdiction allows a pro-rated reduction of taxes, the tax returned will be appropriately reduced. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports.	
EZTaxProRateJCodeEx	BTN, origination and termination information is passed using JCodes. This program takes a J_Code struct as input and calculates taxes for the transaction. This is the same as EZTaxJCodeEx but with percent as input.
EZprivate lineTaxProRateNPANEx	BTN, origination and termination information is passed using NPANXX. This program takes an NPANXX code struct as input and calculates taxes for the transaction. This is the same as EZTaxNPANEx but with input percent
EZTaxProRatePCodeEx	BTN, origination and termination information is passed using PCode. This program takes a PCode struct as input and calculates taxes for the transaction. This is the same as EZTaxPCodeEx but with percent as input.
EZTaxProRateThisJCodeEx	Jurisdiction information is passed using a JCode. This program takes a specific jurisdiction code as input and calculates taxes for the transaction. It is the same as EZTaxThisJCodeEx but with percent as input.
EZTaxProRateThisPCodeEx	Jurisdiction information is passed using a PCode. This program takes a specific jurisdiction code as input and calculates taxes for the transaction. It is the same as EZTaxThisPCodeEx but with percent as input.
EZTaxProRateZipEx	Jurisdiction information is passed using a Zip Code. This program takes a Zip Code plus 4 structure as input and calculates taxes for the transaction. This is the same as EZTaxZipEx but with percent as input.
Private Line or Point-to-Point Functions	
Private line, or Point-to-Point (PTP) as it is often referred to, allows a transaction to be split between the A point and Z point. The percentage specified applies to the A point. The Z point percentage will be automatically calculated as 1-(A pt %).	
EZTaxPrivateLine	Jurisdiction information is passed using a PCode. The program applies the % split to the A point tax calculation, then the remaining % split to the Z point calculation.
EZTaxPrivateLineAdj	Jurisdiction information is passed using a PCode. The program applies the % split to the A point tax adjustment, then the remaining % split to the Z point adjustment.
Miscellaneous Functions	

APIs Listed By Function	
Sorted by Function	Description
The following functions perform miscellaneous operations.	
EZTaxCalcJurisdiction	Returns the JCode that has jurisdiction at the most detailed taxing level.
EZTaxDbVersion	Reads the database version from EZTax.dll and returns it to the caller.
EZTaxDllVersion	Returns the version number of the AFC API.
EZTaxFlushToLogEx	Writes the current internal tax log buffers to the tax log disk file. The internal tax log buffers are normally written to when they are filled to capacity or when EZTaxExitEx is called. This function can be called to force a write to tax log disk file.
EZTaxFreeRates	Frees up the memory from the table of all taxes for a jurisdiction returned from EZTaxGetRates.
EZTaxGroupResults	Sets the mode that AFC uses to group the returned taxes by.
EZTaxSessionDbVersion	Reads the database version being used by the specified AFC Session and returns it to the caller.
EZTaxWriteToLogEx	Allows customers to create the correctly formatted binary AFC log from scratch by passing in data they have stored from a different database or saved from AFC transactions using EZTaxInitExp.
EZTaxWriteToLogV914	Allows customers to create the correctly formatted binary AFC log from scratch by passing in data they have stored from a different database or saved from AFC transactions.

7.8.1 EZTaxAdjDebitJEx

Description

EZTaxAdjDebitJEx accepts transaction data and performs appropriate tax calculations required to refund or credit taxes for Debit transactions. If the AFC Session was started with the tax log open, appropriate tax reductions are logged and will be reflected in tax reports. BTN, origination and termination information is passed using J-codes.

Format

```
short int EZTaxAdjDebitJEx(EZTaxSession session,  
                           struct J_CodeEx *trans,  
                           int discount_type,  
                           int adj_method, int *err_code);
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h
EZTaxStruct.h

Return Value List

[0] : Indicates no tax data available for transaction (not taxable) or an error has occurred
[n] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
*trans : Pointer to data structure which contains transaction data.
J-code transaction information
discount_type : discount type code
adj_method : adjustment method (DLM)
*err_code : error status returned

Remarks

Returns tax adjustments identical to method used for returning tax calculations. Interface data structures are also described in detail in the AFC Data Structures section of the Manual. The AFC calculations section contains a detailed explanation of how taxes are returned to the user.

7.8.2 EZTaxAdjDebitNEx

Description

EZTaxAdjDebitNEx accepts transaction data and performs appropriate tax calculations required to refund or credit taxes for Debit transactionss. If the AFC Session was started with the tax log open, appropriate taxes reductions are logged and will be reflected in tax reports. BTN, origination and termination information is passed using NPANXXs

Format

```
short int EZTaxAdjDebitNEx(EZTaxSession session,  
                           struct NPANXX_codeEx *trans,  
                           int discount_type, int adj_method, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[0]: Indicates no tax data available for transaction (not taxable) or an error has occurred.
[n]: Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session: session handle returned when initialized with call to EZTaxInitEx
*trans: NPANXX transaction information
discount_type: discount type code
adj_method: adjustment method (DLM)
*err_code: error status returned

Remarks

Returns tax adjustments identical to method used for returning tax calculations. Interface data structures are also described in detail in the AFC Data Structures section. The section AFC Calculations explains how taxes are returned to the user in detail.

7.8.3 EZTaxAdjDebitPEx

Description

Function EZTaxAdjDebitPEx accepts transaction data and performs appropriate tax calculations required to refund or credit taxes for Debit transactionss. If the AFC Session was started with the tax log open, appropriate taxes reductions are logged and will be reflected in tax reports. BTN, origination and termination information is passed using PCodes.

Format

```
short int EZTaxAdjDebitPEx(EZTaxSession session,  
                           struct P_CodeEx *trans,  
                           int discount_type, int adj_method, int *err_code)
```

Header(s) Required

EZTaxProto.h

Libraries

EZTax2.lib

Return Value List

0 : Indicates no tax data available for transaction (not taxable) or an error has occurred.

n : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

- session : session handle returned when initialized with call to EZTaxInitEx
- *trans : P-code transaction information
- discount_type : discount type code
- adj_method : adjustment method (DLM)
- *err_code : error status returned

Remarks

Returns tax adjustments identical to method used for returning tax calculations. Interface data structures are also described in detail in the AFC Data Structures section. The section AFC Calculations explains how taxes are returned to the user in detail.

7.8.4 EZTaxAdjDebitTaxInclusiveJCode (EZTaxAdjDebitRevJCode deprecated)

Description

EZTaxAdjDebitTaxInclusiveJCode accepts transaction data and performs a tax inclusive adjustment calculation for a prepaid debit transaction to arrive at the base sale amount. The desired total (charge plus all taxes) is placed in the 'charge' field in the transaction, and base charge necessary to generate the total is returned in the 'base_sale' parameter. Taxes for the desired transaction are returned in the same manner as EZTaxAdjJCodeEx. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using JCodes. The base sale amount will also be stored in the 'optional' field in the log output, converted to an integer.

Format

```
short int EZTaxAdjDebitTaxInclusiveJCode (EZTaxSession session, struct J_CodeEx *trans,  
                                         int discount_type, int adj_method, double *base_sale, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[**-1**]: A critical error occurred while preparing the tax inclusive calculation session
[**0**]: No taxes calculated or an error was found
[**n**]: Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session: session handle returned when initialized with call to EZTaxInitEx
*trans: transaction information
discount_type: discount type code
adj_method: adjustment method (DLM)
*base_sale: calculated charge returned
*err_code: error status returned

Remarks

When both return and err_code are 0, no taxes were found, but there is no error. The base_sale returned will equal the charge sent in. If the charge sent in is insufficient to cover the taxes generated, then no taxes will be calculated and the err_code will be set to -37. An error will be logged to the status file indicating an insufficient charge error occurred.

****Note**:** Non-billable taxes are not included in the calculation of the basesale amount. Adding the base sale amount and all taxes that appear in the log for a transaction may result in a value greater than the target amount; any non-billable taxes returned by the transaction ***will*** be included in the total.

7.8.5 EZTaxAdjDebitTaxInclusiveNPAN (EZTaxAdjDebitRevNPAN deprecated)

Description

EZTaxAdjDebitTaxInclusiveNPAN accepts transaction data and performs a tax inclusive adjustment calculation for a prepaid debit transaction to arrive at the base sale amount. The desired total (charge plus all taxes) is placed in the 'charge' field in the transaction, and base charge necessary to generate that total is returned in the 'base_sale' parameter. Taxes for the desired transaction are returned in the same manner as EZTaxAdjNPANEx. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using NPANXXs. The base sale amount will also be stored in the 'optional' field in the log output, converted to an integer.

Format

```
short int EZTaxAdjDebitTaxInclusiveNPAN(EZTaxSession session,  
struct NPANXX_CodeEx  
*trans, int discount_type, int adj_method, double *base_sale, int *err_code);
```

Header(s) Required

```
EZTaxProto.h  
EZTaxDefine.h  
EZTaxStruct.h
```

Libraries

```
EZTax2.lib
```

Return Value List

[**-1**] : A critical error occurred while preparing the tax inclusive calculation session
[**0**] : No taxes calculated or an error was found
[**n**] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

```
session: session handle returned when initialized with call to EZTaxInitEx  
*trans: transaction information  
discount_type: discount type code  
adj_method: adjustment method (DLM)  
*base_sale: calculated charge returned  
*err_code: error status returned
```

Remarks

When both return and err_code are 0, no taxes were found, but there is no error. The base_sale returned will equal the charge sent in. If the charge sent in is insufficient to cover the taxes generated, then no taxes will be calculated and the err_code will be set to -37. An error will be logged to the status file indicating an insufficient charge error occurred.

****Note**:** Non-billable taxes are not included in the calculation of the base sale amount. Adding the base sale amount and all taxes that appear in the log for a transaction may result in a value greater than the target amount, any non-billable taxes returned by the transaction ***will*** be included in the total.

7.8.6 EZTaxAdjDebitTaxInclusivePCode (EZTaxAdjDebitRevPCode deprecated)

Description

EZTaxAdjDebitTaxInclusivePCode accepts transaction data and performs a tax inclusive adjustment calculation for a prepaid debit transaction to arrive at the base sale amount. The desired total (charge plus all taxes) is placed in the 'charge' field in the transaction, and base charge necessary to generate that total is returned in the 'base_sale' parameter. Taxes for the desired transaction are returned in the same manner as EZTaxAdjPCodeEx. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using PCodes. The base sale amount will also be stored in the 'optional' field in the log output, converted to an integer.

Format

```
short int EZTaxAdjDebitTaxInclusivePCode(EZTaxSession session, struct P_CodeEx *trans,  
                                         int discount_type, int adj_method, double *base_sale, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[-1] : A critical error occurred while preparing the tax inclusive calculation session
[0] : No taxes calculated or an error was found
[n] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session: session handle returned when initialized with call to EZTaxInitEx
*trans: transaction information
discount_type: discount type code
adj_method: adjustment method (DLM)
*base_sale: calculated charge returned
*err_code: error status returned

Remarks

When both return and err_code are 0, no taxes were found, but there is no error. The base_sale returned will equal the charge sent in. If the charge sent in is insufficient to cover the taxes generated, then no taxes will be calculated and the err_code will be set to -37. An error will be logged to the status file indicating an insufficient charge error occurred.

****Note**:** Non-billable taxes are not included in the calculation of the base sale amount. Adding the base sale amount and all taxes that appear in the log for a transaction may result in a value greater than the target amount; any non-billable taxes returned by the transaction ***will*** be included in the total.

7.8.7 EZTaxAdjDebitTaxInclusiveZip (EZTaxAdjDebitRevZip deprecated)

Description

EZTaxAdjDebitTaxInclusiveZip accepts transaction data and performs a tax inclusive adjustment calculation to arrive at the base sale amount. The desired total (charge plus all taxes) is placed in the 'charge' field in the transaction, and base charge necessary to generate that total is returned in the 'base_sale' parameter. Taxes for the desired transaction are returned as EZTaxJCodeEx. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. The base sale amount will also be stored in the 'optional' field in the log output, converted to an integer.

This function passes tax jurisdiction information to AFC by specifying the appropriate zip code, state identifier, county, and locality. AFC uses this information to determine the JCode used for tax calculations. It is important to supply complete address information; there are many duplicate zip codes and localities contained within the AFC database tables. AFC always returns the first match of the data supplied.

Supplying incomplete information can result in tax calculations for an unexpected jurisdiction. When information regarding the locality is not supplied, AFC may be limited to data based upon the county or parish involved. Likewise, when information about the county is not supplied, the AFC engine may be limited to calculations based upon the state and zip code identifiers.

Format

```
short int EZTaxAdjDebitTaxInclusiveZip(EZTaxSession session, struct zip_codeEx *trans,  
                                      int discount_type, int adj_method, double *base_sale, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

- [-1]: A critical error occurred while preparing the tax inclusive calculation session
- [0]: No taxes calculated or an error was found
- [n]: Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session: session handle returned when initialized with call to EZTaxInitEx
*trans: transaction information
discount_type: discount type code
adj_method: adjustment method (DLM)
*base_sale: calculated charge returned
*err_code: error status returned

Remarks

When both return and err_code are 0, no taxes were found, but there is no error. The base_sale returned will equal the charge sent in. If the charge sent in is insufficient to cover the taxes generated, then no taxes will be calculated and the err_code will be set to -37. An error will be logged to the status file indicating an insufficient charge error occurred.

*****Note**: Non-billable taxes are not included in the calculation of the base sale amount. Adding the base sale amount and all taxes that appear in the log for a transaction may result in a value greater than the target amount; any non-billable taxes returned by the transaction *will* be included in the total.***

7.8.8 EZTaxAdjDebitZEx

Description

EZTaxAdjDebitZEx accepts transaction data and performs appropriate tax calculations required to refund or credit taxes for Debit transactionss. If the AFC Session was started with the tax log open, appropriate taxes reductions are logged and will be reflected in tax reports. The tax jurisdiction is specified using address information, identical to EZTaxDebitZEx.

Format

```
short int EZTaxAdjDebitZEx(EZTaxSession session,  
                           struct zip_codeEx *trans,  
                           int discount_type, int adj_method, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[0] : Indicates no tax data available for transaction (not taxable) or an error has occurred.
[n] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
*trans : zip code transaction information
discount_type : discount type code
adj_method : adjustment method (DLM)
* err_code : error status returned

Remarks

Returns tax adjustments identical to method used for returning tax calculations. Interface data structures are also described in detail in the AFC Data Structures section. The section AFC Calculations explains how taxes are returned to the user in detail. AFC currently has Zip Code information for U.S. and Canadian jurisdictions.

7.8.9 EZTaxAdjJCodeEx

Description

EZTaxAdjJCodeEx accepts transaction data and performs appropriate tax calculations required to refund or credit taxes (Debit transactionss use EZTaxAdjDebitEx API's). If the AFC Session was started with the tax log open, appropriate taxes reductions are logged and will be reflected in tax reports. BTN, origination and termination information is passed using JCodes.

Format

```
short int EZTaxAdjJCodeEx(EZTaxSession session, struct J_CodeEx *trans,  
                           int discount_type, int adj_method, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxStruct.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

[0] : Indicates no tax data available for transaction (not taxable) or an error has occurred.
[n] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
*trans : J-code transaction information
discount_type : discount type code
adj_method : adjustment method (DLM)
*err_code : error status returned

Remarks

Returns tax adjustments identical to method used for returning tax calculations. Interface data structures are also described in detail in the AFC Data Structures section. The section AFC Calculations explains how taxes are returned to the user in detail.

7.8.10 EZTaxAdjNPANEx

Description

EZTaxAdjNPANEx accepts transaction data and performs appropriate tax calculations required to refund or credit taxes (Debit transactionss use EZTaxAdjDebitEx API's). If the AFC Session was started with the tax log open, appropriate taxes reductions are logged and will be reflected in tax reports. BTN, origination and termination information is passed using NPANXXs.

Format

```
short int EZTaxAdjNPANEx(EZTaxSession session,  
                           struct NPANXX_codeEx *trans,  
                           int discount_type, int adj_method, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[0] : Indicates no tax data available for transaction (not taxable) or an error has occurred.
[n] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
*trans : NPANXX transaction information
discount_type : discount type code
*err_code : error status returned
adj_method : adjustment method (DLM)

Remarks

Returns tax adjustments identical to method used for returning tax calculations. Interface data structures are also described in detail in the AFC Data Structures section. The section AFC Calculations explains how taxes are returned to the user in detail.

7.8.11 EZTaxAdjPCodeEx

Description

EZTaxAdjPCodeEx accepts transaction data and performs appropriate tax calculations required to refund or credit taxes (Debit transactionss use EZTaxAdjDebitEx API's). If the AFC Session was started with the tax log open, appropriate taxes reductions are logged and will be reflected in tax reports. BTN, origination and termination information is passed using PCodes.

Format

```
short int EZTaxAdjPCodeEx(EZTaxSession session, struct P_CodeEx *trans,  
                           int discount_type, int adj_method, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[0] : Indicates no tax data available for transaction (not taxable) or an error has occurred.
[n] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
*trans : P-code transaction information
discount_type : discount type code
adj_method : adjustment method (DLM)
*err_code : error status returned

Remarks

Returns tax adjustments identical to method used for returning tax calculations. Interface data structures are also described in detail in the AFC Data Structures section. The section AFC Calculations explains how taxes are returned to the user in detail.

7.8.12 EZTaxAdjProRateJCode (EZTaxAdjProRateJCodeEx deprecated)

Description

EZTaxAdjProRateJCode takes a J_Code struct as input and calculates tax adjustments for the transaction. It pro-rates the taxes calculated by the given percentage, for partial-month rates.

The user can select the type of Pro-Rate adjustment handling by using one of the following parameters:

```
prorate_credit_or_cancel = STANDARD_PRORATED_ADJUSTMENT (1)
```

Standard or default processing will not return fixed taxes that are not flagged as pro-ratable. This approach should be used when the desire is to only return taxes that are pro-ratable. An example would be when a client has canceled a service and is due back taxes that are refundable.

```
prorate_credit_or_cancel = CANCEL_PRORATED_CALCULATION (2)
```

Cancel option will return fixed taxes that are not flagged as pro-ratable. This approach should be used when the desire is to cancel out a previous pro-rated calculation. An example would be when a pro-rated tax calculation has been made in error and needs to be canceled out so that the net taxes are 0.

Format

```
short int EZTaxAdjProRateJCode(EZTaxSession session,  
struct J_CodeEx *trans,  
int discount_type, double percent,  
int adj_method, int prorate_credit_or_cancel,  
int *err_code);
```

Return Value List

[0] : Indicates no tax data available for transaction (not taxable) or an error has occurred.

[n] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
*trans : J-code transaction information
discount_type : discount type code
percent : portion of the billing period to pro-rate the tax by
adj_method : adjustment method (DLM)
int prorate_credit_or_cancel : method for how pro-rated adjustment should handle non-prorated taxes
*err_code : error status returned

Remarks

None

7.8.13 EZTaxAdjProRateNPAN (EZTaxAdjProRateNPANEx deprecated)

Description

EZTaxAdjProRateNPAN takes an NPANXX struct as input and calculates tax adjustments for the transaction. It pro-rates the taxes calculated by the given percentage, for partial-month rates.

The user can select the type of Pro-Rate adjustment handling by using one of the following parameters:

`prorate_credit_or_cancel = STANDARD_PRORATED_ADJUSTMENT (1)`

Standard or default processing will not return fixed taxes that are not flagged as pro-ratable. This approach should be used when the desire is to only return taxes that are pro-ratable. An example would be when a client has canceled a service and is due back taxes that are refundable.

`prorate_credit_or_cancel = CANCEL_PRORATED_CALCULATION (2)`

Cancel option will return fixed taxes that are not flagged as pro-ratable. This approach should be used when the desire is to cancel out a previous pro-rated calculation. An example would be when a pro-rated tax calculation has been made in error and needs to be canceled out so that the net taxes are 0.

Format

```
short int EZTaxAdjProRateNPAN (EZTaxSession session,  
struct NPANXX_codeEx *trans,  
int discount_type, double percent,  
int adj_method, int prorate_credit_or_cancel,  
int *err_code);
```

Return Value List

[0] : Indicates no tax data available for transaction (not taxable) or an error has occurred.

[n] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

`session` : session handle returned when initialized with call to EZTaxInitEx
`*trans` : NPANXX transaction information
`discount_type` : discount type code
`percent` : portion of the billing period to pro-rate the tax by
`adj_method` : adjustment method (DLM)
`int prorate_credit_or_cancel` : method for how pro-rated adjustment should handle non-prorated taxes
`*err_code` : error status returned

Remarks

None

7.8.14 EZTaxAdjProRatePCode (EZTaxAdjProRatePCodeEx deprecated)

Description:

EZTaxAdjProRatePCode takes a p-code struct as input and calculates tax adjustments for the transaction. It pro-rates the taxes calculated by the given percentage, for partial-month rates.

The user can select the type of Pro-Rate adjustment handling by using one of the following parameters:

`prorate_credit_or_cancel = STANDARD_PRORATED_ADJUSTMENT (1)`

Standard or default processing will not return fixed taxes that are not flagged as pro-ratable. This approach should be used when the desire is to only return taxes that are pro-ratable. An example would be when a client has canceled a service and is due back taxes that are refundable.

`prorate_credit_or_cancel = CANCEL_PRORATED_CALCULATION (2)`

Cancel option will return fixed taxes that are not flagged as pro-ratable. This approach should be used when the desire is to cancel out a previous pro-rated calculation. An example would be when a pro-rated tax calculation has been made in error and needs to be canceled out so that the net taxes are 0.

Format

```
short int EZTaxAdjProRatePCode (EZTaxSession session,  
                                struct P_CodeEx *trans,  
                                int discount_type, double percent,  
                                int adj_method, int prorate_credit_or_cancel,  
                                int *err_code);
```

Return Value List

[0] : Indicates no tax data available for transaction (not taxable) or an error has occurred.

[n] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

`session` : session handle returned when initialized with call to
`EZTaxInitEx`
`*trans` : p-code transaction information
`discount_type` : discount type code
`percent` : portion of the billing period to pro-rate the tax by
`adj_method` : adjustment method (DLM)
`* int prorate_credit_or_cancel` : method for how pro-rated adjustment should handle non-prorated taxes
`*err_code` : error status returned

Remarks

None

7.8.15 EZTaxAdjProRateThisJCode (EZTaxAdjProRateThisJCodeEx deprecated)

Description

EZTaxAdjProRateThisJCode accepts transaction data and performs appropriate tax calculations. The tax jurisdiction to use for tax calculations is specified to AFC with this method. If the AFC Session was started with the tax log open, appropriate taxes are logged and will be reflected in tax reports. Jurisdiction information is passed using a PCode. It pro-rates the taxes calculated by the given percentage, for partial-month rates.

The user can select the type of Pro-Rate adjustment handling by using one of the following parameters:

prorate_credit_or_cancel = STANDARD_PRORATED_ADJUSTMENT (1)

Standard or default processing will not return fixed taxes that are not flagged as pro-ratable. This approach should be used when the desire is to only return taxes that are pro-ratable. An example would be when a client has canceled a service and is due back taxes that are refundable.

prorate_credit_or_cancel = CANCEL_PRORATED_CALCULATION (2)

Cancel option will return fixed taxes that are not flagged as pro-ratable. This approach should be used when the desire is to cancel out a previous pro-rated calculation. An example would be when a pro-rated tax calculation has been made in error and needs to be canceled out so that the net taxes are 0.

Format

```
short int EZTaxAdjProRateThisJCode (EZTaxSession session,  
struct this_J_Code *trans,  
int discount_type, double percent,  
int adj_method, int prorate_credit_or_cancel,  
int *err_code);
```

Return Value List

[0] : Indicates no tax data available for transaction (not taxable) or an error has occurred.
[n] : Indicates number of taxes stored in the struct taxes_tbl array for this transaction

Parameters

session : Session object returned for session when initialized with call to EZTaxInitEx.

*trans : Pointer to data structure that contains transaction data.

See the this_J_Code struct in EZTaxStruct.h

discount_type : discount type code

percent : portion of the billing period to pro-rate the tax by

adj_method : adjustment method (DLM)

int prorate_credit_or_cancel : method for how pro-rated adjustment should handle non-prorated taxes

*err_code : error status returned

Remarks

This API call is particularly suited for processing local transactions, since there is only one jurisdiction involved.

7.8.16 EZTaxAdjProRateThisPCode (EZTaxAdjProRateThisPCodeEx deprecated)

Description

EZTaxAdjProRateThisPCode accepts transaction data and performs appropriate tax calculations. The tax jurisdiction to use for tax calculations is specified to AFC with this method. If the AFC Session was started with the tax log open, appropriate taxes are logged and will be reflected in tax reports. Jurisdiction information is passed using a PCode. It pro-rates the taxes calculated by the given percentage, for partial-month rates.

The user can select the type of Pro-Rate adjustment handling by using one of the following parameters:

`prorate_credit_or_cancel = STANDARD_PRORATED_ADJUSTMENT (1)`

Standard or default processing will not return fixed taxes that are not flagged as pro-ratable. This approach should be used when the desire is to only return taxes that are pro-ratable. An example would be when a client has canceled a service and is due back taxes that are refundable.

`prorate_credit_or_cancel = CANCEL_PRORATED_CALCULATION (2)`

Cancel option will return fixed taxes that are not flagged as pro-ratable. This approach should be used when the desire is to cancel out a previous pro-rated calculation. An example would be when a pro-rated tax calculation has been made in error and needs to be canceled out so that the net taxes are 0.

Format

```
short int EZTaxAdjProRateThisPCode (EZTaxSession session,  
struct this_P_Code *trans,  
int discount_type, double percent,  
int adj_method, int prorate_credit_or_cancel,  
int *err_code);
```

Return Value List

[0] : Indicates no tax data available for transaction (not taxable) or an error has occurred.

[n] : Indicates number of taxes stored in the struct taxes_tbl array for this transaction

Parameters

`session` : Session object returned for session when initialized with call to EZTaxInitEx.

`*trans` : Pointer to data structure that contains transaction data.

See the `this_P_Code` struct in `EZTaxStruct.h`

`discount_type` : discount type code

`percent` : portion of the billing period to pro-rate the tax by

`adj_method` : adjustment method (DLM)

`int prorate_credit_or_cancel` : method for how pro-rated adjustment should handle non-prorated taxes

`*err_code` : error status returned

Remarks

This API call is particularly suited for processing local transactions, since there is only one jurisdiction involved.

7.8.17 EZTaxAdjProRateZip (EZTaxAdjProRateZipEx deprecated)

Description

EZTaxAdjProRateZip takes a p-code struct as input and calculates tax adjustments for the transaction. It pro-rates the taxes calculated by the given percentage, for partial-month rates.

The user can select the type of Pro-Rate adjustment handling by using one of the following parameters:

`prorate_credit_or_cancel = STANDARD_PRORATED_ADJUSTMENT (1)`

Standard or default processing will not return fixed taxes that are not flagged as pro-ratable. This approach should be used when the desire is to only return taxes that are pro-ratable. An example would be when a client has canceled a service and is due back taxes that are refundable.

`prorate_credit_or_cancel = CANCEL_PRORATED_CALCULATION (2)`

Cancel option will return fixed taxes that are not flagged as pro-ratable. This approach should be used when the desire is to cancel out a previous pro-rated calculation. An example would be when a pro-rated tax calculation has been made in error and needs to be canceled out so that the net taxes are 0.

Format

```
short int EZTaxAdjProRateZip (EZTaxSession session,  
                           struct zip_codeEx *trans,  
                           int discount_type, double percent,  
                           int adj_method, int adj_method, int prorate_credit_or_cancel,  
                           int *err_code);
```

Return Value List

[0] : Indicates no tax data available for transaction (not taxable) or an error has occurred.

[n] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

`session` : session handle returned when initialized with call to EZTaxInitEx
`*trans` : zip code transaction information
`discount_type` : discount type code
`percent` : portion of the billing period to pro-rate the tax by
`adj_method` : adjustment method (DLM)
`int prorate_credit_or_cancel` : method for how pro-rated adjustment should handle non-prorated taxes
`*err_code` : error status returned

Remarks

None

7.8.18 EZTaxAdjTaxInclusiveJCode (EZTaxAdjRevJCode deprecated)

Description

EZTaxAdjTaxInclusiveJCode accepts transaction data and performs a tax inclusive adjustment calculation to arrive at the base sale amount. The desired total (charge plus all taxes) is placed in the 'charge' field in the transaction, and base charge necessary to generate that total is returned in the 'base_sale' parameter. Taxes for the desired transaction are returned in the same manner as EZTaxAdjJCodeEx does. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using JCodes. The base sale amount will also be stored in the 'optional' field in the log output, converted to an integer.

Format

```
short int EZTaxAdjTaxInclusiveJCode(EZTaxSession session, struct J_CodeEx *trans,  
                                    int discount_type, int adj_method, double *base_sale, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[**-1**]: A critical error occurred while preparing the tax inclusive calculation session
[**0**]: No taxes calculated or an error was found
[**n**]: Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session: session handle returned when initialized with call to EZTaxInitEx
*trans: transaction information
discount_type: discount type code
adj_method: adjustment method (DLM)
*base_sale: calculated charge returned
*err_code: error status returned

Remarks

When both return and err_code are 0, no taxes were found, but there is no error. The base_sale returned will equal the charge sent in. If the charge sent in is insufficient to cover the taxes generated, then no taxes will be calculated and the err_code will be set to -37. An error will be logged to the status file indicating an insufficient charge error occurred.

****Note**:** Non-billable taxes are not included in the calculation of the base sale amount. Adding the base sale amount and all taxes that appear in the log for a transaction may result in a value greater than the target amount, as any non-billable taxes returned by the transaction ***will*** be included in that total.

7.8.19 EZTaxAdjTaxInclusiveNPAN (EZTaxAdjRevNPAN deprecated)

Description

EZTaxAdjTaxInclusiveNPAN accepts transaction data and performs a tax inclusive adjustment calculation to arrive at the base sale amount. The desired total (charge plus all taxes) is placed in the 'charge' field in the transaction, and base charge necessary to generate that total is returned in the 'base_sale' parameter. Taxes for the desired transaction are returned in the same manner as EZTaxAdjNPANEx does. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using NPANXXs. The base sale amount will also be stored in the 'optional' field in the log output, converted to an integer.

Format

```
short int EZTaxAdjTaxInclusiveNPAN (EZTaxSession session, struct NPANXX_CodeEx *trans,  
                                    int discount_type, int adj_method, double *base_sale, int *err_code);
```

Header(s) Required

```
EZTaxProto.h  
EZTaxDefine.h  
EZTaxStruct.h
```

Libraries

```
EZTax2.lib
```

Return Value List

- [**-1**]: A critical error occurred while preparing the tax inclusive calculation session
- [**0**]: No taxes calculated or an error was found
- [**n**]: Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

- session**: session handle returned when initialized with call to EZTaxInitEx
- *trans**: transaction information
- discount_type**: discount type code
- adj_method**: adjustment method (DLM)
- *base_sale**: calculated charge returned
- *err_code**: error status returned

Remarks

When both return and err_code are 0, no taxes were found, but there is no error. The base_sale returned will equal the charge sent in. If the charge sent in is insufficient to cover the taxes generated, then no taxes will be calculated and the err_code will be set to -37. An error will be logged to the status file indicating an insufficient charge error occurred.

****Note**:** Non-billable taxes are not included in the calculation of the base sale amount. Adding the base sale amount and all taxes that appear in the log for a transaction may result in a value greater than the target amount, as any non-billable taxes returned by the transaction *will* be included in that total.

7.8.20 EZTaxAdjTaxInclusivePCode (EZTaxAdjRevPCode deprecated)

Description

EZTaxAdjTaxInclusivePCode accepts transaction data and performs a tax inclusive adjustment calculation to arrive at the base sale amount. The desired total (charge plus all taxes) is placed in the 'charge' field in the transaction, and base charge necessary to generate that total is returned in the 'base_sale' parameter. Taxes for the desired transaction are returned in the same manner as EZTaxAdjPCodeEx does. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using PCodes. The base sale amount will also be stored in the 'optional' field in the log output, converted to an integer.

Format

```
short int EZTaxAdjTaxInclusivePCode(EZTaxSession session, struct P_CodeEx *trans,  
                                    int discount_type, int adj_method, double *base_sale, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[**-1**]: A critical error occurred while preparing the tax inclusive calculation session
[**0**]: No taxes calculated or an error was found
[**n**]: Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session: session handle returned when initialized with call to EZTaxInitEx
*trans: transaction information
discount_type: discount type code
adj_method: adjustment method (DLM)
*base_sale: calculated charge returned
*err_code: error status returned

Remarks

When both return and err_code are 0, no taxes were found, but there is no error. The base_sale returned will equal the charge sent in. If the charge sent in is insufficient to cover the taxes generated, then no taxes will be calculated and the err_code will be set to -37. An error will be logged to the status file indicating an insufficient charge error occurred.

****Note**:** Non-billable taxes are not included in the calculation of the base sale amount. Adding the base sale amount and all taxes that appear in the log for a transaction may result in a value greater than the target amount, as any non-billable taxes returned by the transaction ***will*** be included in that total.

7.8.21 EZTaxAdjTaxInclusiveZip (EZTaxAdjRevZip deprecated)

Description

EZTaxAdjTaxInclusiveZip accepts transaction data and performs a tax inclusive adjustment calculation to arrive at the base sale amount. The desired total (charge plus all taxes) is placed in the 'charge' field in the transaction, and base charge necessary to generate that total is returned in the 'base_sale' parameter. Taxes for the desired transaction are returned as EZTaxJCodeEx does. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. The base sale amount will also be stored in the 'optional' field in the log output, converted to an integer.

When the user knows the tax jurisdiction, address information can be supplied to specify it. This function passes tax jurisdiction information to AFC by specifying the appropriate zip code, state identifier, county, and locality. AFC uses this information to determine the JCode used for tax calculations. It is important to supply complete address information as there are many duplicate zip codes and localities contained within the AFC database tables. AFC always returns the first match of the data supplied.

Supplying incomplete information can result in tax calculations for an unexpected jurisdiction. When information regarding the locality is not supplied, AFC may be limited to data based upon the county or parish involved. Likewise, when information about the county is not supplied, the AFC engine may be limited to performance of calculations based upon the state and zip code identifiers.

Format

```
short int EZTaxAdjTaxInclusiveZip(EZTaxSession session, struct zip_codeEx *trans,  
                                  int discount_type, int adj_method, double *base_sale, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

- [-1]: A critical error occurred while preparing the tax inclusive calculation session
- [0]: No taxes calculated or an error was found
- [n]: Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session: session handle returned when initialized with call to EZTaxInitEx
*trans: transaction information
discount_type: discount type code
adj_method: adjustment method (DLM)
*base_sale: calculated charge returned
*err_code: error status returned

Remarks

When both return and err_code are 0, no taxes were found, but there is no error. The base_sale returned will equal the charge sent in. If the charge sent in is insufficient to cover the taxes generated, then no taxes will be calculated and the err_code will be set to -37. An error will be logged to the status file indicating an insufficient charge error occurred.

*****Note**:*** *Non-billable taxes are not included in the calculation of the base sale amount. Adding the base sale amount and all taxes that appear in the log for a transaction may result in a value greater than the target amount, as any non-billable taxes returned by the transaction *will* be included in that total.*

7.8.22 EZTaxAdjTPPEx

Description

EZTaxAdjTPPEx accepts Tangible Personal Property (TPP) transaction data and performs appropriate tax calculations required to refund or credit taxes. The user provides up to three addresses: Ship From, Ship To and Point of Acceptance. Rules will be applied against the input addresses to determine in what jurisdiction any SALES or USE taxation would be applicable. A Nexus Table may optionally be provided; if so it will be checked to determine if nexus exists in the state found to have taxing jurisdiction. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports.

Format

```
short int EZTaxAdjTPPEx(EZTaxSession session,
                         struct TPP_addrEx *tpp_trans, int discount_type,
                         struct nexus_table *nex_tab,
                         short int nexus_count, int adj_method, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[0]: Indicates no tax data available for transaction (not taxable) or an error has occurred.
[n]: Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
*tpp_trans : transaction information
discount_type : discount type code
*nex_tab : the nexus table
nexus_count : count of the entries in the nexus table
adj_method : adjustment method (DLM)
*err_code : error status returned

Remarks

Nexus Table

The Nexus Table (see structure definition below) provides a list of USPS state abbreviations, each with a flag indicating whether the user has nexus in that state. This table is optional. If the table is not provided, nexus is assumed in all states. The user is responsible for establishing this table and passing it to EZtax.

Interface data structures are also described in detail in the AFC Data Structures section. The section AFC Calculations explains how taxes are returned to the user in detail.

Minimum Requirements

Ship From address must be provided. Address structure must include COUNTRY ISO, STATE, LOCALITY, and ZIP CODE. All fields in any unused address must be NULL.

If only the Ship From address is provided, the transaction will always be INTRASTATE, and Ship From will be the taxing jurisdiction. If only Ship From and Point-of-Acceptance addresses are given, Point-of-Acceptance is used as the Ship To address.

The Transaction Type and Service Type fields of the tax_data structure are set within this function. Any value previously set will be ignored.

In addition to other error codes that may be set for AFC API function calls, the following in particular apply to EZTaxAdjTPPEx-

- 27 Missing or Invalid Ship From Address
- 28 Invalid Ship To Address
- 29 Invalid Point-of-Acceptance
- 30 Invalid State Input

7.8.23 EZTaxAdjZipEx

Description

EZTaxAdjZipEx accepts transaction data and performs appropriate tax calculations required to refund or credit taxes for U.S. jurisdictions (Debit transactionss use EZTaxAdjDebitEx API's). If the AFC Session was started with the tax log open, appropriate taxes reductions are logged and will be reflected in tax reports. The tax jurisdiction is specified using address information, identical to EZTaxZipEx.

Format

```
short int EZTaxAdjZipEx(EZTaxSession session, struct zip_codeEx  
                        *trans, int discount_type, int adj_method, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[0] : Indicates no tax data available for transaction (not taxable) or an error has occurred.
[n] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
*trans : zip code transaction information
discount_type : discount type code
adj_method : adjustment method (DLM)
*err_code : error status returned

Remarks

This API is used for tax adjustments for U.S. jurisdictions.

Returns tax adjustments identical to method used for returning tax calculations. Interface data structures are also described in detail in the AFC Data Structures section. The section AFC Calculations explains how taxes are returned to the user in detail.

7.8.24 EZTaxCalcJurisdiction

Description

EZTaxCalcJurisdiction returns the JCode that has jurisdiction at the most detailed taxing level.

Format

```
unsigned long int EZTaxCalcJurisdiction (EZTaxSession session,  
                                     struct J_CodeEx *trans, int *err_code)
```

Header(s) Required

EZTaxStruct.h
EZTaxDefine.h
EZTaxProto.h

Libraries

EZTax2.lib

Return Value List

The most detailed jurisdiction determined to be applicable based on the transaction information; however, if there is an error, both JCODE_NOT_FOUND (-11) and err_code zero are returned.

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
*trans : J-code transaction information

Remarks

This API call does not log the taxes or calculate and return a tax table. It does apply the jurisdiction rules to determine what taxing authority has jurisdiction for that transaction.

7.8.25 EZTaxClearExclusion

Description

EZTaxClearExclusion is used to remove all exclusions that have been set.

Format

```
short int EZTaxClearExclusion(EZTaxSession session);
```

Header(s) Required

EZTaxProto.h

Libraries

EZTax2.lib

Return Value List

TRUE : was successful.

FALSE : failed.

Parameters

Session : session handle returned when initialized with call to EZTaxInitEx

Remarks

None

7.8.26 EZTaxClearTSR

Description

EZTaxClearTSR allows API functionality to delete the data created by EZTaxGetTSR. This is used to prevent memory leaks during operation.

Format

```
void EZTaxClearTSR(EZTaxSession session, struct rtr_data* rpt)
```

Return Value List

Void

Parameters

session: session handle returned for session when initialized with call to EZTaxInitEx

rpt: pointer to the T/S Report created by EZTaxGetTSR. This will be cleared from memory by EZTaxClearTSR.

Remarks

None

7.8.27 EZTaxClose

Description

Function EZTaxClose is utilized to close the NPANXX, address and zip code files.

Format

```
short int EZTaxClose(long int hdl, short int file);
```

Header(s) Required

EZTaxProto.h
EZTaxStruct.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

always returns TRUE

Parameters

hdl : Session handle returned when initialized with call to EZTaxInit
file : database to open (not used)

Remarks

This API call is obsolete and provided only for backwards compatibility. EZTaxExitSession and EZTaxExitSessionEx will now close all databases that are still open.

7.8.28 EZTaxCountryToPCode

Description

EZTaxCountryToPCode takes a 3 byte country ISO code and converts it to the correct PCode for that country.

Format

```
unsigned long int EZTaxCountryToPCode(EZTaxSession session,  
                                     const char* iso_code,  
                                     int *err_code)
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

PCode : converted from the input fips code
0 : if conversion is unsuccessful

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
*iso_code: 3 byte country ISO code to convert
*err_code : error status returned

Remarks

The value “UNS” will return the PCode for unsupported country. If the initial call fails due to the county code not being supported, this can be a fallback value to use.

7.8.29 EZTaxDbVersion

Description

Function EZTaxDbVersion reads the database version from EZTax.dll and returns it to the caller.

Format

```
short int EZTaxDbVersion(char *psz_EZTax_dat, char *psz_db_version);
```

Header(s) Required

EZTaxDefine.h,
EZTaxProto.h

Libraries

EZTax2.lib

Return Value List

TRUE if database version is returned in psz_db_version

FALSE if database version couldn't be obtained

Parameters

*psz_EZTax_dat : path to EZTax.dat file

*psz_db_version : address of character array that will receive the database version.

Remarks

The database version will be returned in the form ww.xx.yy.zz, where each piece of the database version number can be 1 or 2 digits.

7.8.30 EZTaxDebitJEx

Description

EZTaxDebitJEx accepts transaction data and performs appropriate tax calculations. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using JCodes.

Format

```
short int EZTaxDebitJEx(EZTaxSession session, struct J_CodeEx *trans,  
                        int *err_code)
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[0] : Indicates no tax data available for transaction (not taxable) or an error has occurred.
[n] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
*trans : J-code transaction information
*err_code : error status returned

Remarks

Interface data structures are also described in detail in the AFC Data Structures section. The section AFC Calculations explains how taxes are returned to the user in detail.

Returns tax adjustments identical to method used for returning tax calculations.

7.8.31 EZTaxDebitNEx

Description

EZTaxDebitNEx accepts transaction data and performs appropriate tax calculations. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using NPANXXs.

Format

```
short int EZTaxDebitNEx(EZTaxSession session,  
                         struct NPANXX_codeEx *trans,  
                         int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[0] : Indicates no tax data available for transaction (not taxable) or an error has occurred.
[n] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
*trans : NPANXX transaction information
*err_code : error status returned

Remarks

Interface data structures are also described in detail in the AFC Data Structures section. The section AFC Calculations explains how taxes are returned to the user in detail.

7.8.32 EZTaxDebitPEx

Description

EZTaxDebitPEx accepts transaction data and performs appropriate tax calculations. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using PCodes.

Format

```
short int EZTaxDebitPEx(EZTaxSession session, struct P_CodeEx *trans,  
                        int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[0] : Indicates no tax data available for transaction (not taxable) or an error has occurred.
[n] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
*trans : P-code transaction information
*err_code : error status returned

Remarks

Interface data structures are also described in detail in the AFC Data Structures section. The section AFC Calculations explains how taxes are returned to the user in detail.

7.8.33 EZTaxDebitTaxInclusiveJCode (EZTaxDebitRevJCode deprecated)

Description

EZTaxDebitTaxInclusiveJCode accepts transaction data and performs a tax inclusive calculation for prepaid debit transactions to arrive at the base sale amount. The desired total (charge plus all taxes) is placed in the 'charge' field in the transaction, and base charge necessary to generate that total is returned in the 'base_sale' parameter. Taxes for the desired transaction are returned in the same manner as EZTaxJCodeEx. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using JCodes. The base sale amount will also be stored in the 'optional' field in the log output, converted to an integer.

Format

```
short int EZTaxDebitTaxInclusiveJCode(EZTaxSession session, struct J_CodeEx *trans,  
                                     double *base_sale, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[**-1**]: A critical error occurred while preparing the tax inclusive calculation session
[**0**] : No taxes calculated or an error was found
[**n**]: Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session: session handle returned when initialized with call to EZTaxInitEx
***trans:** transaction information
***base_sale:** calculated charge returned
***err_code:** error status returned

Remarks

When both return and err_code are 0, no taxes were found, but there is no error. The base_sale returned will equal the charge sent in. If the charge sent in is insufficient to cover the taxes generated, then no taxes will be calculated and the err_code will be set to -37. An error will be logged to the status file indicating an insufficient charge error occurred.

****Note**:** Non-billable taxes are not included in the calculation of the base sale amount. Adding the base sale amount and all taxes that appear in the log for a transaction may result in a value greater than the target amount; any non-billable taxes returned by the transaction *will* be included in the total.

7.8.34 EZTaxDebitTaxInclusiveNPAN (EZTaxDebitRevNPAN deprecated)

Description

EZTaxDebitTaxInclusiveNPAN accepts transaction data and performs a tax inclusive calculation for prepaid debit transactions to arrive at the base sale amount. The desired total (charge plus all taxes) is placed in the 'charge' field in the transaction, and base charge necessary to generate that total is returned in the 'base_sale' parameter. Taxes for the desired transaction are returned in the same manner as EZTaxNPANEx. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using NPANXXs. The base sale amount will also be stored in the 'optional' field in the log output, converted to an integer.

Format

```
short int EZTaxDebitTaxInclusiveNPAN(EZTaxSession session, struct NPANXX_CodeEx *trans,  
double *base_sale, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[**-1**]: A critical error occurred while preparing the tax inclusive calculation session
[**0**]: No taxes calculated or an error was found
[**n**]: Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session: session handle returned when initialized with call to EZTaxInitEx
***trans**: transaction information
***base_sale**: calculated charge returned
***err_code**: error status returned

Remarks

When both return and err_code are 0, no taxes were found, but there is no error. The base_sale returned will equal the charge sent in. If the charge sent in is insufficient to cover the taxes generated, then no taxes will be calculated and the err_code will be set to -37. An error will be logged to the status file indicating an insufficient charge error occurred.

****Note**:** Non-billable taxes are not included in the calculation of the base sale amount. Adding the base sale amount and all taxes that appear in the log for a transaction may result in a value greater than the target amount; any non-billable taxes returned by the transaction ***will*** be included in the total.

7.8.35 EZTaxDebitTaxInclusivePCode (EZTaxDebitRevPCode deprecated)

Description

EZTaxDebitTaxInclusivePCode accepts transaction data and performs a tax inclusive calculation for prepaid debit transactions to arrive at the base sale amount. The desired total (charge plus all taxes) is placed in the 'charge' field in the transaction, and base charge necessary to generate that total is returned in the 'base_sale' parameter. Taxes for the desired transaction are returned in the same manner as EZTaxPCodeEx. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using PCodes. The base sale amount will also be stored in the 'optional' field in the log output, converted to an integer.

Format

```
short int EZTaxDebitTaxInclusivePCode(EZTaxSession session, struct P_CodeEx *trans,  
double *base_sale, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[**-1**]: A critical error occurred while preparing the tax inclusive calculation session
[**0**]: No taxes calculated or an error was found
[**n**]: Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session: session handle returned when initialized with call to EZTaxInitEx
***trans**: transaction information
***base_sale**: calculated charge returned
***err_code**: error status returned

Remarks

When both return and err_code are 0, no taxes were found, but there is no error. The base_sale returned will equal the charge sent in. If the charge sent in is insufficient to cover the taxes generated, then no taxes will be calculated and the err_code will be set to -37. An error will be logged to the status file indicating an insufficient charge error occurred.

****Note**:** Non-billable taxes are not included in the calculation of the base sale amount. Adding the base sale amount and all taxes that appear in the log for a transaction may result in a value greater than the target amount; any non-billable taxes returned by the transaction ***will*** be included in the total.

7.8.36 EZTaxDebitTaxInclusiveZip (EZTaxDebitRevZip deprecated)

Description

EZTaxDebitTaxInclusiveZip accepts transaction data and performs a tax inclusive calculation for prepaid debit transactions to arrive at the base sale amount. The desired total (charge plus all taxes) is placed in the 'charge' field in the transaction, and base charge necessary to generate that total is returned in the 'base_sale' parameter. Taxes for the desired transaction are returned in the same manner as EZTaxJCodeEx. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. The base sale amount will also be stored in the 'optional' field in the log output, converted to an integer.

This function passes tax jurisdiction information to AFC by specifying the appropriate zip code, state identifier, county, and locality. AFC uses this information to determine the JCode used for tax calculations. It is important to supply complete address information; there are many duplicate zip codes and localities contained within the AFC database tables. AFC always returns the first match of the data supplied.

Supplying incomplete information can result in tax calculations for an unexpected jurisdiction. When information regarding the locality is not supplied, AFC may be limited to data based upon the county or parish involved. Likewise, when information about the county is not supplied, the EZtax engine may be limited to performance of calculations based upon the state and zip code identifiers.

Format

```
short int EZTaxDebitTaxInclusiveZip(EZTaxSession session, struct zip_codeEx *trans,  
                                    double *base_sale, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

- [-1]: A critical error occurred while preparing the tax inclusive calculation session
- [0]: No taxes calculated or an error was found
- [n]: Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session: session handle returned when initialized with call to EZTaxInitEx
*trans: transaction information
*base_sale: calculated charge returned
*err_code: error status returned

Remarks

When both return and err_code are 0, no taxes were found, but there is no error. The base_sale returned will equal the charge sent in. If the charge sent in is insufficient to cover the taxes generated,

then no taxes will be calculated and the err_code will be set to -37. An error will be logged to the status file indicating an insufficient charge error occurred.

*****Note**: Non-billable taxes are not included in the calculation of the base sale amount. Adding the base sale amount and all taxes that appear in the log for a transaction may result in a value greater than the target amount; any non-billable taxes returned by the transaction *will* be included in the total.***

7.8.37 EZTaxDllVersion

Description

Function EZTaxDllVersion returns the version number of the AFC API.

Format

```
short int EZTaxDllVersion(char *psz_library_version);
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h

Libraries

EZTax2.lib

Return Value List

TRUE : if the database version is returned in psz_library_version

FALSE : if database version couldn't be obtained

Parameters

psz_library_version : version string returned

Remarks

The version string is the same value that is displayed in the EZTax.sta status file.

[7.8.38 EZTaxExit](#)

Description

EZTaxExit writes the contents of database buffers to disk, de-allocates memory, closes databases and performs other clean-up duties which free up resources utilized by EZtax. This function will then exit the AFC system.

EZTaxExit terminates all sessions that were created with calls to EZTaxInit in addition to processes mentioned above.

Format

```
short int EZTaxExit(EZTaxSession session)
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

always returns TRUE

Parameters:

Void

Remarks

EZTaxExit must be called to terminate all existing AFC Session(s).

EZTaxExit will close any sessions still running.

7.8.39 EZTaxExitSessionEx

Description

EZTaxExitSessionEx will terminate a session specified by valid session handle. EZTaxExitSessionEx writes the current internal tax log buffers to the tax log disk file, de-allocates cached database memory, closes all AFC database files, and performs other clean-up duties which free up resources utilized for the specified session.

Format

```
short int EZTaxExitSessionEx(EZTaxSession session);
```

Header(s) Required

EZTaxProto.h

Libraries

EZTax2.lib

Return Value List

TRUE : if session exited successfully

FALSE : if an error occurred

Parameters

session : session handle returned when initialized with call to EZTaxInitEx

Remarks

The session handle and the struct taxes_tbl * returned by EZTaxInitEx during the initialization of the given session handle are invalidated by a call to this function. Attempting to use either after a call to EZTaxExitSessionEx will cause errors.

7.8.40 EZTaxFlushToLogEx

Description

EZtax FlushToLogEx writes the current internal tax log buffers to the tax log disk file. The internal tax log buffers are normally written to when they are filled to capacity or EZTaxExit is called. This function can be called to force a write to tax log disk file.

Format

```
short int EZTaxFlushToLogEx(EZTaxSession session);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

always returns TRUE

Parameters

session : session handle returned when initialized with call to EZTaxInitEx

Remarks

EZTaxFlushToLogEx writes the content of internal tax log buffers to disk.

7.8.41 EZTaxFreeRates

Description

EZTaxFreeRates frees up the memory from the table of all taxes for a jurisdiction returned from EZTaxGetRates

Format

```
short int EZTaxFreeRates(struct jurisdictionTaxes *taxes);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

always returns 0

Parameters

*taxes : tax information to be freed

Remarks

None

7.8.42 EZTaxFtoPCodeEx

Description

EZTaxFtoPCodeEx converts a Fips Code into a PCode. Zero is returned if not found.

Format

```
unsigned long int EZTaxFtoPCodeEx(EZTaxSession session, char *Fips,  
int *err_code)
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

PCode : converted from the input fips code
0 : if conversion is unsuccessful

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
*Fips: Fips code to convert
*err_code : error status returned

Remarks

None

7.8.43 EZTaxGetAddressEx

Description

EZTaxGetAddressEx utilizes a J-Code as input to produce the location/address data for the jurisdiction identified by the J-Code.

Format

```
short int EZTaxGetAddressEx(EZTaxSession session,  
                           unsigned long int j_code,  
                           struct address_data_p4 *address,  
                           int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

NOT_FOUND : EZTaxGetAddressEx failed due to invalid JCode or an error has occurred.
FOUND : EZTaxGetAddressEx successfully retrieved address
MORE : EZTaxGetAddressEx successfully retrieved address and other valid addresses exist for this jurisdiction. Other address can be obtained by calling EZTaxNextAddressEx.

Parameters

Session : session handle returned when initialized with call to EZTaxInitEx
j_code : j-code to find an address for
*address : the address information returned
*err_code : error status code

Remarks

None

7.8.44 EZTaxGetCustomLog

Description

EZTaxGetCustomLog is a routine that returns a pointer to a log structure, which is used for customized logging.

Format

```
struct EZTax_logEx *EZTaxGetCustomLog(EZTaxSession session)
```

Return Value List

NULL : Indicates an error has occurred
pointer : The logging structure pointer

Parameters

session: session handle returned with call to initialize the AFC Session.

Remarks

This function is used after a call to initialize the AFC Session and before any taxes are calculated. The log structure returned can be used to call EZTaxWriteToLogEx to output the information to the AFC log. EZTaxGetCustomLogCount can be used to get the record count in the log table as needed.

When a transaction has been processed, the custom log structure will contain the tax information generated by that transaction.

The value in eztax_logEx.p_code is the jcode representation of the jurisdiction.

7.8.45 EZTaxGetCountryID

Description

EZTaxGetCountryID returns the country ID associated with the PCode passed in.

For a list of country codes, refer to the **NpaNxx Country Id** column listed under “Appendix D – International NPANXX IDs” in document *TM_00561_AFC Comprehensive Guide to Reports and Data Files.pdf*.

Format

```
unsigned int EZTaxGetCountryID(EZTaxSession session, unsigned long int PCode, int *err_code);
```

Return Value List

unsigned int representing the country id associated with the PCode passed in.

Parameters

session : Session object returned for session when initialized with call to EZTaxInitEx.

PCode : AFC proprietary code representing a taxing jurisdiction

*err_code: Error codes returned. See the header EZTaxDefine.h for definitions.

Remarks

None

7.8.46 EZTaxGetCustomLogCount

Description

Function returns the number of records in the current custom log table. This function can be used in conjunction with EZTaxGetCustomLog() which returns a pointer to the custom log table.

Format

```
short int EZTaxGetCustomLogCount(EZTaxSession session)
```

Return Value List

Count of records : 0 if no records exist, else the number of records in the custom log table.

Parameters

session: session handle returned with call to initialize the AFC Session.

Remarks

GetCustomLogCount returns the number of taxes contained in the EZTaxGetCustomLog structure. The log count includes no summarization. The log count always includes the non-billable tax count.

7.8.47 EZTaxGetHandle

Description

EZTaxGetHandle returns the handle object given the session.

Format

```
long int EZTaxGetHandle(EZTaxSession session);
```

Header(s) Required

EZTaxProto.h

EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

Handle : value if found

[-1] : otherwise

Parameters

session: session handle returned when initialized with call to EZTaxInitEx

Remarks

None

7.8.48 EZTaxGetJurisdiction

Description

EZTaxGetJurisdiction retrieves the jurisdiction where the transaction is determined to be taxed at. Current use is in the batch executables that report tax revenue group by state.

Format

```
unsigned long int EZTaxGetJurisdiction(EZTaxSession session,  
                                     struct J_CodeEx trans, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

[0] : if jurisdiction is not found
JCcode : otherwise

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
trans : transaction information
*err_code : error status returned

Remarks

None

7.8.49 EZTaxGetLogName

Description

EZTaxGetLogName retrieves the log filename when it is specified using the filelocs.txt. Current use is in the batch executables that open the log file and must know the filename.

Format

```
short int EZTaxGetLogName(EZTaxSession session, char *filename);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

SESSION_NOT_INIT : if session handle is invalid

TRUE : otherwise

Parameters

session: session handle returned when initialized with call to EZTaxInitEx

*filename: filename from filelocs.txt configuration file or paths variable EZTaxInitEx was called with

Remarks

None

7.8.50 EZTaxGetLogV914

Description

EZTaxGetLogV914 is a routine that returns a pointer to a log structure, which is used for customized logging.

Format

```
struct eztax_log_v914 *EZTaxGetLogV914(EZTaxSession session)
```

Return Value List

NULL : Indicates an error has occurred
pointer : The v914 logging structure pointer

Parameters

session: session handle returned with call to initialize the AFC Session.

Remarks

This function is used after a call to initialize the AFC Session and before any taxes are calculated. The log structure returned can be used to call EZTaxWriteToLogV914 to output the information to the AFC log. EZTaxGetLogV914Count can be used to get the record count in the log table.

When a transaction has been processed, the V914 tax log structure will contain the tax information generated by that transaction. In addition it will contain the charge and t/s pair used in the tax calculation. This can be valuable when processing bundled transactions, as the t/s pair will be the bundled product and the charge the % allocated to this t/s pair.

The value in eztax_log_v914.p_code is the jcode representation of the jurisdiction.

7.8.51 EZTaxGetLogV914Count

Description

Function returns the number of records in the current custom log table. This function can be used in conjunction with EZTaxGetLogV914 () which returns a pointer to the v9114 tax log table.

Format

```
short int EZTaxGetLogV914Count (EZTaxSession session)
```

Return Value List

Count of records : 0 if no records exist, else the number of records in the v914 tax log table.

Parameters

session: session handle returned with call to initialize the AFC Session.

Remarks

GetLogV914Count returns the number of taxes contained in the EZTaxGetLogV914 structure. The log count includes no summarization. The log count always includes the non-billable tax count.

7.8.52 EZTaxGetRates

Description

EZTaxGetRates utilizes a PCode as input to produce the table of all taxes for that jurisdiction.

Format

```
short int EZTaxGetRates(EZTaxSession session, unsigned long int p_code,  
struct jurisdictionTaxes *taxes, int *err_code)
```

Header (s) Required

EZTaxProto.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

Success: 0
Failure: 1

Parameters

pCode : jurisdiction to get rate information for
*taxes : tax rates
*err_code : error codes

Remarks

None

7.8.53 EZTaxGetSession

Description

EZTaxGetSession returns the session object given the handle.

Format

```
EZTaxSession EZTaxGetSession(long int hdl)
```

Header (s) Required

EZTaxProto.h

EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

returns the session pointer

Parameters

hdl : the session handle

Remarks

None

7.8.54 EZTaxGetStateID

Description

EZTaxGetStateID returns the state id associated with the PCode passed in.

For a list of US state codes, refer to the **Id** column listed under “Appendix B – Supported States, Territories and Provinces” in document *TM_00561_AFC Comprehensive Guide to Reports and Data Files.pdf*.

Format

```
unsigned int EZTaxGetStateID (EZTaxSession session, unsigned long int PCode, int *err_code);
```

Return Value List

unsigned int representing the state id associated with the PCode passed in.

Parameters

session : Session object returned for session when initialized with call to EZTaxInitEx.

PCode : AFC proprietary code representing a taxing jurisdiction

*err_code: Error codes returned. See the header EZTaxDefine.h for definitions.

Remarks

None

7.8.55 EZTaxGetTaxCatV98

Description

EZTaxGetTaxCatV98 retrieves the tax category description for a specified tax code. Current use is in utilities reporting from the log, but users can also use this new feature.

Format

```
char *EZTaxGetTaxCatV98(EZTaxSession session, int taxCode);
```

Header (s) Required

EZTaxProto.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

the tax category description

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
taxCode : the tax type to get the description for

Remarks

None

7.8.56 EZTaxGetTaxDescription

Description

EZTaxGetTaxDescription retrieves the tax description for a specified tax code. Current use is in utilities reporting from the log, but users can also use this new feature.

Format

```
char *EZTaxGetTaxDescription(EZTaxSession session, int taxCode);
```

Header (s) Required

EZTaxProto.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

the tax description

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
taxCode : the tax type to get the description for

Remarks

None

7.8.57 EZTaxGetTSR

Description

EZTaxGetTSR allows applications API functionality to get the transaction service report. A flag is used to specify whether the report data should be returned or written to file.

Format

```
struct rtr_data* EZTaxGetTSR(EZTaxSession session, int *size, short int returnData)
```

Return Value List

Pointer to rtr_data : if method succeeded AND returnData is true

NULL : otherwise

Parameters

session: session handle returned with call to initialize the AFC Session.

size : an integer passed by reference that will be set to the size of the returned array

returnData : flag that determines if the TSR data should be returned or written to file.

Remarks

EZTaxGetTSR returns a pointer to an array of data. This data is cleared from memory using EZTaxClearTSR. Using EZTaxGetTSR without calling EZTaxClearTSR can lead to memory leaks.

If returnData is false, the function flushes the TSR data to file and clears the data from memory.

If returnData is true, the function returns the TSR data as an array and does not clear the data. Once the returned data is processed and no longer needed, the calling application should call EZTaxClearTSR() to clear the data from memory.

7.8.58 EZTaxGroupResults

Description

This function sets the mode that AFC uses to group the returned taxes by. After calling this function, the results in the tax return table will be grouped as specified by the groupMode parameter.

Format

```
short int EZTaxGroupResults(EZTaxSession session,  
                           unsigned long int groupMode);
```

Header (s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

short int : TRUE or FALSE indicating if the function succeeded or not.

Parameters:

int groupMode: Constant indicating how tax calculation results should be returned. The options to group taxes by tax level are following:

- GROUP_SAME_LEVEL: Indicates that taxes at the same level should be grouped together.
- GROUP_CO_LOCAL: When using this option, AFC will group all state taxes into a single record, and county and local taxes will be grouped together into a separate record.
- GROUP_ST_CO_LOCAL. State, county and local taxes will be grouped together.

The options to group sales taxes separately may be appended to a tax level option by using the bitwise OR operator (|). These options are:

- GROUP_SALES: Sales taxes (tax type 1) and Use Taxes (tax type 49) will be grouped separately from other taxes.
- GROUP_SALES_CATEGORY: All taxes in the sales tax category will be grouped into a single record.

The DEFAULT option may be used to disable tax grouping. These options are included in the EZTaxDefine.h header file.

7.8.59 EZTaxInitDirEx

Description

EZTaxInitDirEx reads AFC files and initializes the AFC system. Multiple calls may be made to set up completely independent sessions.

Format

```
struct enhanced_taxes_tbl *EZTaxInitDirEx(short int tax_log,  
    struct file_path *paths,  
    EZTaxSession *session,  
    char *directory);
```

Header(s) Required

EZTaxProto.h
EZTaxStruct.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

pointer : to structure where taxes are returned later
NULL : if an error occurred

Parameters

tax_log : flag to log taxes or not
*paths : data structure with file-paths, if null or incomplete, paths are read from file_locs.txt
*session : session handle returned
*directory : name of directory that AFC will search for EZTax.cfg and filelocs.txt

Remarks

This function initializes and starts an AFC Session.

Multiple sessions can be initialized by successive calls to EZTaxInitDirEx.

7.8.60 EZTaxInitEx

Description

EZTaxInitEx reads AFC files and initializes the AFC system. Multiple calls may be made to set up completely independent sessions.

Format

```
struct enhanced_taxes_tbl *EZTaxInitEx(short int tax_log,  
                                      struct file_path *paths,  
                                      EZTaxSession *session);
```

Header(s) Required

EZTaxProto.h
EZTaxStruct.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

pointer : to structure where taxes are returned later
NULL : if an error occurred

Parameters

tax_log : flag to log taxes or not
*paths : data structure with file-paths, if null or incomplete, paths
are read from file_locs.txt
*session : session handle returned

Remarks

This function initializes and starts an AFC Session.

Multiple sessions can be initialized by successive calls to EZTaxInitEx.

7.8.61 EZTaxInitExMT

Description

EZTaxInitExMT reads AFC files and initializes the AFC system. Multiple calls may be made to set up completely independent sessions.

Format

```
struct enhanced_taxes_tbl *EZTaxInitEx(short int tax_log,  
                                      struct file_path *paths,  
                                      EZTaxSession *session,  
                                      char *directory);
```

Header(s) Required

EZTaxProto.h
EZTaxStruct.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

pointer : to structure where taxes are returned later
NULL : if an error occurred

Parameters

tax_log : flag to log taxes or not
*paths : data structure with file-paths, if null or incomplete, paths
are read from file_locs.txt
*session : session object returned
*directory : name of directory that AFC will search for EZTax.cfg and filelocs.txt

Remarks

This function initializes and starts an AFC Session.

This function is designed to be thread-safe - other sessions on other threads can be running AFC calculations when this function is called.

Be sure to call EZTaxExitSessionEx for each EZTaxSession object created by EZTaxInitExMT. EZTaxExit will not close sessions created by EZTaxInitExMT. Failure to call EZTaxExitSessionEx will result in memory leaks.

7.8.62 EZTaxInitV914

Description

EZTaxInitV914 reads AFC files and initializes the AFC system. Multiple calls may be made to set up completely independent sessions.

The input structure allows all optional input files to be specified through the initialization method. This method does not use filelocs.txt and will ignore exclusion, nexus and bundle file settings in EZTax.cfg.

Format

```
struct enhanced_taxes_tbl * EZTaxInitV914(short int tax_log_enabled,  
                                         struct file_path_v914 *paths,  
                                         EZTaxSession *session);
```

Return Value List

pointer : to structure where taxes are returned later
NULL : if an error occurred

Parameters

- *tax_log_enabled : flag to enable the log taxes process
- *paths : data structure with file-paths.
- *session : session object returned
- *directory : name of directory that AFC will search for EZTax.cfg and filelocs.txt

Remarks

This function initializes and starts an AFC Session and is designed to be thread-safe - other sessions on other threads can be running AFC calculations when this function is called.

Be sure to call EZTaxExitSessionEx for each EZTaxSession object created by EZTaxInitV914. EZTaxExit will not close sessions created by EZTaxInitV914. Failure to call EZTaxExitSessionEx will result in memory leaks.

7.8.63 EZTaxInitV98

Description

EZTaxInitV98 reads AFC files and initializes the AFC system. It differs from EZTaxInitExMT in that the tax table returned includes tax category information. Multiple calls may be made to set up completely independent sessions.

Format

```
struct taxes_tbl_v98 *EZTaxInitV98(short int tax_log,  
                                     struct file_path *paths,  
                                     EZTaxSession *session,  
                                     char *directory);
```

Header(s) Required

EZTaxProto.h
EZTaxStruct.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

pointer : to structure where taxes are returned later
NULL : if an error occurred

Parameters

tax_log : flag to log taxes or not
*paths : data structure with file-paths, if null or incomplete, paths are read from file_locs.txt
*session : session object returned
*directory : name of directory that AFC will search for EZTax.cfg and filelocs.txt

Remarks

This function initializes and starts an AFC Session.

This function is designed to be thread-safe - other sessions on other threads can be running AFC calculations when this function is called.

****Note**:** Be sure to call EZTaxExitSessionEx for each EZTaxSession object created by EZTaxInitV98. EZTaxExit will not close sessions created by EZTaxInitV98. Failure to call EZTaxExitSessionEx will result in memory leaks.

7.8.64 EZTaxJCodeEx

Description

EZTaxJCodeEx accepts transaction data and performs appropriate tax calculations. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using JCodes.

Format

```
short int EZTaxJCodeEx(EZTaxSession session, struct J_CodeEx *trans,  
                      int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[0] : No taxes calculated or an error was found
[n] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session : session handle returned when initialized with call to EZTaxInitEx
*trans : transaction information
*err_code : error status returned

Remarks

When both return and err_code are 0, no taxes were found, but there is no error.

7.8.65 EZTaxJtoPCodeEx

Description

EZTaxJtoPCodeEx Returns PCode that is cross-referenced to a JCode.

Format

```
unsigned long int EZTaxJtoPCodeEx(EZTaxSession session,  
                                  unsigned long int JCode,  
                                  int *err_code);
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

PCode : for the input jcode if found, 0 otherwise

Parameters

session: Session handle returned when initialized with call to EZTaxInitEx
JCode: j-code to convert
*err_code: Error status returned

Remarks

None

7.8.66 EZTaxJTypeEx (EZTaxJType deprecated)

Description

Returns transaction type for a given pair of JCodes indicating whether the call is interstate or intrastate. The first JCode is the originating and the second the terminating JCode. Will set error code to INTERSTATE_NOT_VALID_FOR_INTERNATIONAL_CALLS

If the originating country and terminating country are not the same.

The returned value can also be used to determine service types. For example for transaction type, 19, 20, 21, 59, the interstate service type is 49, the intrastate service is 50. For transaction type 61, the interstate service is 585, intrastate service is 586. But caller is responsible to map INTERSTATE or INTRASTATE to the right service type.

Format

```
short int EZTaxJTypeEx(EZTaxSession session,  
                      unsigned long int orig_J_Code,  
                      unsigned long int term_J_Code,  
                      int* err_code)
```

Return Value List

INTERSTATE or INTRASTATE : depending on the origination and termination States.

Parameters

session: Session handle returned when session was initialized
orig_J_Code: origination j-code
term_J_Code: termination j-code
*err_code : error status returned

Remarks

This call will eventually be deprecated. Recommend usage of EZTaxPTTypeEx(...) which works with PCodes in place of this call.

7.8.67 EZTaxJurisdiction

Description

EZTaxJurisdiction returns the first JCode at the lowest level.

Format

```
unsigned long int EZTaxJurisdiction(long int hdl,  
                                     struct J_Code *trans);
```

Header(s) Required

EZTaxStruct.h
EZTaxDefine.h
EZTaxProto.h

Libraries

EZTax2.lib

Return Value List

[0] : if no j-code found.
JCode : otherwise as jurisdiction was found.

Parameters

hdl : Session handle returned when initialized with call to EZTaxInit
*trans : transaction information

Remarks

None

7.8.68 EZTaxMaxTaxCount

Description

EZTaxMaxTaxCount returns the maximum number of taxes that one transaction can generate.

Format

```
Short int EZTaxMaxTaxCount();
```

Header(s) Required

EZTaxProto.h

Libraries

EZTax2.lib

Return Value List

Count : Maximum tax count generated by one transaction

Parameters

None

Remarks

None

7.8.69 EZTaxNextAddressEx

Description

EZTaxNextAddressEx returns the next sequential record containing alternative address information for a specific JCode. This function can be used to obtain additional address for a specified jurisdiction when EZTaxGetAddressEx returns MORE, indicating additional information is available. The majority of jurisdictions have more than one set of address/zip code data associated with them.

Format

```
short int EZTaxNextAddressEx(EZTaxSession session,  
                           struct address_data_p4 *address,  
                           int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

FOUND : is returned if last address found
MORE : is returned if an address is found and more addresses are available
NOT_FOUND : is returned if not found

Parameters

session : session handle returned when initialized with call to
EZTaxInitEx
*address : address information
*err_code : error status returned

Remarks

None

7.8.70 EZTaxNextCustomerEx

Description

EZTaxNextCustomerEx puts the taxes calculated in the customer table into a customer tax table that the client can reference and reinitializes the customer table to prepare for the next set of customer records.

Format

```
int EZTaxNextCustomerEx(EZTaxSession session);
```

Libraries

EZTax2.lib

Header(s) Required

EZTaxProto.h,
EZTaxDefine.h,
EZTaxStruct.h

Return Value List

Tax count : positive if all taxes waiting have been returned
[-1] : if there was an error
[-1*tax count] : if more taxes are waiting to be retrieved

Parameters

session : Session handle returned for session when initialized with call to EZTaxInit.

References:

EZTaxSetInvoiceModeEx
EZTaxSetInvoiceModeV98

Remarks

To be used if the system is in Invoice Mode. Reinitializes the customer table to prepare for the next set of customer records when all waiting taxes have been retrieved. If the return value is negative (and not -1), EZTaxNextCustomerEx must be called again to retrieve more taxes. Interface data structures are also described in detail in the AFC Data Structures section.

****Note**:** The summarized taxes returned when calling EZTaxNextCustomerEx are identical to the total of the individual taxes returned from the taxing APIs. Typically one or the other is used but not both. Combining the individual taxes and the summarized taxes will result in twice as much taxes as should be normal.

7.8.71 EZTaxNPANEx

Description

Function EZTaxNPANEx takes an NPANXX_codeEx struct as input and calculates taxes for the transaction. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using NPANXXs.

Format

```
short int EZTaxNPANEx(EZTaxSession session,  
                      struct NPANXX_codeEx *trans,  
                      int *err_code);
```

Header(s) Required

EZTaxProto.h,
EZTaxDefine.h,
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[n] : number of taxes returned

Parameters

session : Session handle returned when initialized with call to EZTaxInitEx
*trans : npanxx transaction information
*err_code : Error status returned

Remarks

Interface data structures are also described in detail in the AFC Data Structures section. The section AFC Calculations explains how taxes are returned to the user in detail.

7.8.72 EZTaxNtoJCodeEx

Description

EZTaxNtoJCodeEx returns JCode that is cross-referenced to an NPANXX

Format

```
unsigned long int EZTaxNtoJCodeEx (EZTaxSession session, unsigned long  
int npanxx, int *err_code);
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

FALSE (0) : Indicates that the PCode specified was not found or an error has occurred.
JCode : Jurisdiction code

Parameters

session : Session handle returned when initialized with call to EZTaxInitEx
npanxx : 6 digit NPANXX (area code and exchange)
*err_code : Error codes returned. See the header EZTaxDefine.h for definitions.

Remarks

This function converts an NPANXX into a J-Code. If the supplied NPANXX was not found or an error has occurred 0 is returned, which is valid U.S. Federal level code. Please check the error code before using the JCode that is returned.

7.8.73 EZTaxNTTypeEx

Description

EZTaxNTTypeEx returns transaction type for a given pair of NPANXX's indicating whether the call is interstate or intrastate. The first NPANXX is the originating and the second is the terminating NPANXX.

Format

```
short int EZTaxNTTypeEx(EZTaxSession session,  
                        unsigned long int orig_NPANXX,  
                        unsigned long int term_NPANXX);
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[n] : 1 if the call was Interstate, 2 if the call was Intrastate

Parameters

session: Session handle returned when initialized with call to EZTaxInitEx
orig_NPANXX : A NPANXX for the originating part of the call.
term_NPANXX : A NPANXX for the terminating part of the call.

Remarks

This program takes the origination and termination NPANXXs as input and determines if the call is INTERSTATE or INTRASTATE and returns the corresponding response to the caller

7.8.74 EZTaxOldOvrJCode

Description

EZTaxOldOvrJCode specifies an override for a specific tax. Tax jurisdiction information is passed via jurisdiction code. It takes a J_Code and an override structure as input and inserts a tax override into the AFC db system.

Format

```
short int EZTaxOvrJCode(EZTaxSession session, unsigned long int j_code,  
                        struct tax_ovrd *over);
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

FALSE : Override failed
TRUE : Override established

Parameters

session : Session object returned for session when initialized with call to EZTaxInitEx or EZTaxInitExMT.
j_code : Specifies jurisdiction to override via JCode.
over : Pointer to structure that specifies specific tax to override. See struct tax_ovrd in EZTaxStruct.h for detailed information. Interface data structures are also described in detail in the AFC Data Structures section.

Remarks

Taxes which are overridden are in effect from the time the override is established until the AFC Session is terminated. All overrides can be removed from the system using the method EZTaxRestore.

Overrides must be approached with caution. The system allows overrides of all taxes and allows the user to specify the scope of the override. Overriding a tax on a state or county level can impact taxes in excess of a thousand jurisdictions in some cases. Further detail is contained in the AFC Data Structure section.

7.8.75 EZTaxOvrJCodeEx

Description

EZTaxOvrJCodeEx specifies an override for a specific tax. Tax Jurisdiction information is passed via jurisdiction code. It takes a J_Code and an override structure as input and inserts a tax override into the AFC db system.

Format

```
short int EZTaxOvrJCodeEx(EZTaxSession session,  
                           unsigned long int j_code,  
                           struct enhancedOverride *over)
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Parameters

session : Session object returned for session when initialized with call to EZTaxInitEx
j_code : Specifies jurisdiction to override via JCode
*over : pointer to override information in the same form as returned by EZTaxGetRates. See struct enhancedOverride in EZTaxStruct.h for detailed information.

Note: This structure points to other structures with detailed information on effective date for each tax rate and tax bracket. Interface data structures are also described in detail in the AFC Data Structures section.

Return Value List

FALSE : Override failed
TRUE : Override established

Remarks

Taxes which are overridden are in effect from the time the override is established until the AFC Session is terminated. All overrides can be removed from the system using the method EZTaxRestore.

Overrides must be approached with caution. The system allows overrides of all taxes and allows the user to specify the scope of the override. Overriding a tax on a state or county level can impact taxes in excess of a thousand jurisdictions in some cases. Further detail is contained in the AFC Data Structure section.

7.8.76 EZTaxOvrNPANEx

Description

EZTaxOvrNPANEx specifies an override for a specific tax. Tax jurisdiction information is passed via NPANXX. It takes an NPANXX and an override structure as input and inserts a tax override into the AFC db system.

Format

```
short int EZTaxOvrNPANEx(EZTaxSession session,  
                           unsigned long int npanxx,  
                           struct enhancedOverride *over)
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Parameters

session : Session object returned for session when initialized with call to EZTaxInitEx
npanxx : Specifies jurisdiction to override via NPANXX
*over : pointer to override information in the same form as returned by EZTaxGetRates. See struct enhancedOverride in EZTaxStruct.h for detailed information.

Note: This structure points to other structures with detailed information on effective date for each tax rate and tax bracket. Interface data structures are also described in detail in the AFC Data Structures section.

Return Value List

FALSE : Override failed
TRUE : Override established

Remarks

Taxes which are overridden are in effect from the time the override is established until the AFC Session is terminated. All overrides can be removed from the system using the method EZTaxRestore.

Overrides must be approached with caution. The system allows overrides of all taxes and allows the user to specify the scope of the override. Overriding a tax on a state or county level can impact taxes in excess of a thousand jurisdictions in some cases. Further detail is contained in the AFC Data Structure section.

7.8.77 EZTaxOvrPCodeEx

Description

EZTaxOvrPCodeEx specifies an override for a specific tax. Tax jurisdiction information is passed via permanent jurisdiction code. It takes a P_Code and an override structure as input and inserts a tax override into the AFC db system.

Format

```
short int EZTaxOvrPCodeEx(EZTaxSession session,  
                           unsigned long int p_code,  
                           struct enhancedOverride *over)
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Parameters

session : Session object returned for session when initialized with call to EZTaxInitEx
p_code : Specifies jurisdiction to override via PCode
*over : pointer to override information in the same form as returned by EZTaxGetRates. See struct enhancedOverride in EZTaxStruct.h for detailed information.

Note: This structure points to other structures with detailed information on effective date for each tax rate and tax bracket. Interface data structures are also described in detail in the AFC Data Structures section.

Return Value List

FALSE : Override failed
TRUE : Override established

Remarks

Taxes which are overridden are in effect from the time the override is established until the AFC Session is terminated. All overrides can be removed from the system using the method EZTaxRestore.

Overrides must be approached with caution. The system allows overrides of all taxes and allows the user to specify the scope of the override. Overriding a tax on a state or county level can impact taxes in excess of a thousand jurisdictions in some cases. Further detail is contained in the AFC Data Structure section.

7.8.78 EZTaxOvrZipEx

Description

EZTaxOvrZipEx specifies an override for a specific tax. Tax jurisdiction information is passed via zip code and address. It takes zip code plus 4 and address information, as well as an override structure as input and inserts a tax override into the AFC db system.

Format

```
short int EZTaxOvrZipEx(EZTaxSession session,
                         struct zip_address_p4 int zip_addr,
                         struct enhancedOverride *over,
                         int *err_code)
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h

Libraries

EZTax2.lib

Parameters

session : Session object returned for session when initialized with call to EZTaxInitEx.
zip_addr : Specifies jurisdiction to override via Zip code and address.
*over : pointer to override information in the same form as returned by EZTaxGetRates. See struct enhancedOverride and struct zip_address_p4 in EZTaxStruct.h for detailed information.
err_code : error status returned

Note: The enhancedOverride structure points to other structures with detailed information on effective date for each tax rate and tax bracket. Interface data structures are also described in detail in the AFC Data Structures section.

Return Value List

FALSE : Override failed
TRUE : Override established

Remarks

Taxes which are overridden are in effect from the time the override is established until the AFC Session is terminated. All overrides can be removed from the system using the method EZTaxRestore.

Overrides must be approached with caution. The system allows overrides of all taxes and allows the user to specify the scope of the override. Overriding a tax on a state or county level can impact taxes in excess of a thousand jurisdictions in some cases. Further detail is contained in the AFC Data Structure section.

7.8.79 EZTaxPCodeEx

Description

EZTaxPCodeEx accepts transaction data and performs appropriate tax calculations. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using PCodes. This program takes a PCode struct as input and calculates taxes for the transaction.

Format

```
short int EZTaxPCodeEx(EZTaxSession session,  
                      struct P_CodeEx *trans,  
                      int* err_code);
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

FALSE : Indicates no tax data available for transaction (not taxable) or an error has occurred.
[n] : Indicates number of taxes stored in the struct taxes_tbl array for this transaction.

Parameters

session : Session object returned for session when initialized with call to EZTaxInitEx.
*trans : Pointer to data structure that contains transaction data. See the P_CodeEx struct in the header file EZTaxStruct.h.
*err_code : Error status returned

Remarks

Interface data structures are also described in detail in the AFC Data Structures section. The section AFC Calculations explains how taxes are returned to the user in detail.

7.8.80 EZTaxPrivateLine

Description

EZTaxPrivateLine accepts transaction data and performs tax calculations for a private line transaction. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. This program takes a private_line_P_Code struct as input and calculates taxes for the transaction. The private_line_P_Code struct has fields named point_A (origination point), point_Z(termination point), both in PCode, and split which is a number between 0 and 1 to indicate the percentage which applies to the origination point. Any remaining charges are then applied to the termination point.

Format

```
short int EZTaxPrivateLine (EZTaxSession session,  
                           struct private_line_P_Code *trans,  
                           int *err_code);
```

Return Value List

- [0] : No taxes calculated or an error was found
- [n] : Indicates number of taxes stored in the struct taxes_tbl array for this transaction

Parameters

- session : Session handle returned when initialized with call to EZTaxInitEx
- *trans : Pointer to data structure that contains transaction data. See the private_line_P_Code struct in the header file EZTaxStruct.h.
- *err_code : Error status returned

7.8.81 EZTaxPrivateLineAdj

Description

EZTaxPrivateLineAdj accepts transaction data and performs tax adjustments for a private line transaction. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. This program takes a private_line_P_Code struct as input and calculates taxes for the transaction. The private_line_P_Code struct has fields named point_A (origination point), point_Z (termination point), both in PCode, and split which is a number between 0 and 1 to indicate the percentage which applies to the origination point. Any remaining charges are then applied to the termination point.

Format

```
short int EZTaxPrivateLineAdj (EZTaxSession session,  
    struct private_line_P_Code *trans,  
    int discount_type,  
    int adj_method,  
    int *err_code);
```

Return Value List

- [0] : No taxes calculated or an error was found
- [n] : Indicates number of taxes stored in the struct taxes_tbl array for this transaction

Parameters

session : Session handle returned when initialized with call to EZTaxInitEx
*trans : Pointer to data structure that contains transaction data. See the private_line_P_Code struct in the header file EZTaxStruct.h.
discount_type : discount type code
adj_method : adjustment method (DLM)
*err_code : Error status returned

Remarks

Returns tax adjustments identical to method used for returning tax calculations. Interface data structures are also described in detail in the AFC Data Structures section. The section AFC Calculations explains how taxes are returned to the user in detail.

7.8.82 EZTaxProRateJCodeEx

Description

EZTaxProRateJCodeEx accepts transaction data and performs appropriate tax calculations. If the taxing jurisdiction allows a pro-rated reduction of taxes, the tax returned will be appropriately reduced. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using JCodes. This program takes a J_Code struct as input and calculates taxes for the transaction. This is the same as EZTaxJCodeEx but with percent as input.

Format

```
short int EZTaxProRateJCodeEx(EZTaxSession session,
                               struct J_CodeEx *trans,
                               double percent, int *err_code);
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[0] : No taxes calculated or an error was found
[n] : Indicates number of taxes stored in the struct taxes_tbl array for this transaction

Parameters

session : Session handle returned when initialized with call to EZTaxInitEx
*trans : Pointer to data structure that contains transaction data. See the J_CodeEx struct in the header file EZTaxStruct.h.
percent : percentage to pro-rate
*err_code : Error status returned

Remarks

Some taxes can be pro-rated when a customer receives only a partial period of service. The transaction data should be filled in with the data for a complete period, and the percent value calculated as the fraction of a period. When both return and *err_code are 0, no taxes were found, but there is no error.

7.8.83 EZTaxProRateNPANEx

Description

EZTaxProRateNPANEx accepts transaction data and performs appropriate tax calculations. If the taxing jurisdiction allows a pro-rated reduction of taxes, the tax returned will be appropriately reduced. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using NPANXX. This program takes an NPANXX_code struct as input and calculates taxes for the transaction. This is the same as EZTaxNPANEx but with input percent.

Format

```
short int EZTaxProRateNPANEx(EZTaxSession session,
                           struct NPANXX_codeEx *trans,
                           double percent, int *err_code);
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[0]: No taxes calculated or an error was found
[n]: Indicates number of taxes stored in the struct taxes_tbl array for this transaction

Parameters

session : Session handle returned when initialized with call to EZTaxInitEx
*trans : Pointer to data structure that contains transaction data. See the NPANXX_codeEx struct in the header file EZTaxStruct.h.
percent : percentage to pro-rate
*err_code : Error status returned

Remarks

Some taxes can be pro-rated when a customer receives only a partial period of service. The transaction data should be filled in with the data for a complete period, and the percent value calculated as the fraction of a period. When both return and *err_code are 0, no taxes were found, but there is no error.

7.8.84 EZTaxProRatePCodeEx

Description

EZTaxProRatePCodeEx accepts transaction data and performs appropriate tax calculations. If the taxing jurisdiction allows a pro-rated reduction of taxes, the tax returned will be appropriately reduced. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using PCode. This program takes a PCode struct as input and calculates taxes for the transaction. This is the same as EZTaxPCodeEx but with percent as input.

Format

```
short int EZTaxProRatePCodeEx(EZTaxSession session,  
                           struct P_CodeEx *trans,  
                           double percent, int *err_code);
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h

Libraries

EZTax2.lib

Return Value List

[0]: No taxes calculated or an error was found
[n]: Indicates number of taxes stored in the struct taxes_tbl array for this transaction

Parameters

session : Session handle returned when initialized with call to EZTaxInitEx
*trans : Pointer to data structure that contains transaction data. See the P_CodeEx struct in the header file EZTaxStruct.h.
percent : percentage to pro-rate
*err_code : Error status returned

Remarks

Some taxes can be pro-rated when a customer receives only a partial period of service. The transaction data should be filled in with the data for a complete period, and the percent value calculated as the fraction of a period. When both return and *err_code are 0, no taxes were found, but there is no error.

7.8.85 EZTaxProRateThisJCodeEx

Description

EZTaxProRateThisJCodeEx accepts transaction data and performs appropriate tax calculations. If the taxing jurisdiction allows a pro-rated reduction of taxes, the tax returned will be appropriately reduced. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. Jurisdiction information is passed using a JCode. This program takes a specific jurisdiction code as input and calculates taxes for the transaction. It is the same as EZTaxThisJCodeEx but with input percent.

Format

```
short int EZTaxProRateThisJCodeEx(EZTaxSession session,  
                                 struct this_J_CodeEx *trans,  
                                 double percent, int *err_code);
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[0]: No taxes calculated or an error was found
[n]: Indicates number of taxes stored in the struct taxes_tbl array for this transaction

Parameters

session : Session handle returned when initialized with call to EZTaxInitEx
*trans : Pointer to data structure that contains transaction data. See the this_J_CodeEx struct in the header file EZTaxStruct.h.
percent : percentage to pro-rate
*err_code : Error status returned

Remarks

Some taxes can be pro-rated when a customer receives only a partial period of service. The transaction data should be filled in with the data for a complete period, and the percent value calculated as the fraction of a period. When both return and *err_code are 0, no taxes were found, but there is no error.

7.8.86 EZTaxProRateThisPCodeEx

Description

EZTaxProRateThisPCodeEx accepts transaction data and performs appropriate tax calculations. If the taxing jurisdiction allows a pro-rated reduction of taxes, the tax returned will be appropriately reduced. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. Jurisdiction information is passed using a PCode. This program takes a specific jurisdiction code as input and calculates taxes for the transaction. It is the same as EZTaxThisPCodeEx but with input percent.

Format

```
short int EZTaxProRateThisPCodeEx(EZTaxSession session,
                                  struct this_P_CodeEx *trans,
                                  double percent, int *err_code);
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[0]: No taxes calculated or an error was found
[n] : Indicates number of taxes stored in the struct taxes_tbl array for this transaction

Parameters

session : Session handle returned when initialized with call to EZTaxInitEx
*trans : Pointer to data structure that contains transaction data. See the this_P_CodeEx struct in the header file EZTaxStruct.h.
percent : percentage to pro-rate
*err_code : Error status returned

Remarks

Some taxes can be pro-rated when a customer receives only a partial period of service. The transaction data should be filled in with the data for a complete period, and the percent value calculated as the fraction of a period. When both return and *err_code are 0, no taxes were found, but there is no error.

7.8.87 EZTaxProRateZipEx

Description

EZTaxProRateZipEx accepts transaction data and performs appropriate tax calculations. If the taxing jurisdiction allows a pro-rated reduction of taxes, the tax returned will be appropriately reduced. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. Jurisdiction information is passed using a Zip Code. This program takes a zip code plus 4 structure as input and calculates taxes for the transaction. This is the same as EZTaxZipEx but with percent as input.

Format

```
short int EZTaxProRateZipEx(EZTaxSession session,
                           struct zip_codeEx *trans,
                           double percent, int *err_code);
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h

Libraries

EZTax2.lib

Return Value List

[0]: No taxes calculated or an error was found
[n]: Indicates number of taxes stored in the struct taxes_tbl array for this transaction

Parameters

session : Session handle returned when initialized with call to EZTaxInitEx
*trans : Pointer to data structure that contains transaction data. See the zip_codeEx struct in the header file EZTaxStruct.h.
percent : percentage to pro-rate
*err_code : Error status returned

Remarks

Some taxes can be pro-rated when a customer receives only a partial period of service. The transaction data should be filled in with the data for a complete period, and the percent value calculated as the fraction of a period. When both return and *err_code are 0, no taxes were found, but there is no error.

7.8.88 EZTaxPtoFipsEx

Description

EZTaxPtoFipsEx returns FIPS Code that is cross-referenced to a PCode. It converts a PCode into a Fips Code. Zero is returned if not found.

Format

```
char *EZTaxPtoFipsEx(EZTaxSession session,  
                      unsigned long int PCode,  
                      int *err_code);
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

FALSE : (0) Indicates that the PCode specified was not found or an error has occurred.
FIPS Code : FIPS Jurisdiction code

Parameters

session : Session handle returned when initialized with call to EZTaxInitEx.
PCode : Permanent jurisdiction code.
*err_code : Error codes returned. See the header EZTaxDefine.h for definitions.

Remarks

If the supplied PCode was not found or an error has occurred 0 is returned.

7.8.89 EZTaxPtoJCodeEx

Description

EZTaxPtoJCodeEx converts a PCode into a J-Code. PCODE_NOT_FOUND returned if not found.

Format

```
unsigned long int EZTaxPtoJCodeEx(EZTaxSession session, unsigned long  
int PCode, int *err_code);
```

Header(s) Required

EZTaxDefine.h
EZTaxStruct.h
EZTaxProto.h

Libraries

EZTax2.lib

Return Value List

FALSE (0) : Indicates that the PCode specified was not found or an error has occurred.
JCode : Jurisdiction code

Parameters

session : Session handle returned when initialized with call to EZTaxInitEx
PCode : Permanent jurisdiction code.
*err_code : Error codes returned. See the header EZTaxDefine.h for definitions.

Remarks

If the supplied PCode was not found or an error has occurred 0 is returned, which is a valid U.S. Federal level code.

Please check the error code before using the JCode that is returned.

7.8.90 EZTaxPTTypeEx

Description

Returns transaction type for a given pair of PCodes indicating whether the call is interstate or intrastate. The first JCode is the originating and the second the terminating PCode. Will set error code to INTERSTATE_NOT_VALID_FOR_INTERNATIONAL_CALLS

If the originating country and terminating country are not the same.

The returned value can also be used to determine service types. For example for transaction type, 19, 20, 21, 59, the interstate service type is 49, the intrastate service is 50. For transaction type 61, the interstate service is 585, intrastate service is 586. But caller is responsible to map INTERSTATE or INTRASTATE to the right service type.

Format

```
short int EZTaxPTType(EZTaxSession session,  
                      unsigned long int orig_PCode,  
                      unsigned long int term_PCode,  
                      int* err_code)
```

Return Value List

INTERSTATE or INTRASTATE : depending on the origination and termination States.

Parameters

session: Session handle returned when session was initialized
orig_PCode: origination p-code
term_PCode: termination p-code
*err_code : error status returned

Remarks

This is the preferred API to be used and replaces usage of EZTaxJTTType and EZTaxJTTTypeEx.

7.8.91 EZTaxRestoreEx

Description

EZTaxRestoreEx removes all overrides that have been inserted, by the user, into the AFC databases. This action occurs automatically on EZTaxExit. The function provides the user with the ability to remove prior overrides, without terminating an AFC Session.

This program restores the AFC db to its original state (i.e. before overrides). All tables modified, except for the federal tax table, are restored by restoring EZTax.dll and clearing cache. This is so because all tables are modified by appending overrides to the end of the table and changing the references from the old record to the new record. Therefore, when cache is cleared and new data is pulled from the unmodified EZTax.dll, it references the original, unmodified record in the associated tables. As federal taxes do not work in this fashion, federal tax overrides are installed and restored using a unique procedure.

Format

```
short int EZTaxRestoreEx(EZTaxSession session);
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

TRUE : Restore succeeded
FALSE : Restore failed

Parameters

session : Session object returned for session when initialized with call to EZTaxInitEx.

Remarks

All prior overrides are lost and the system databases are returned to their original configuration.

7.8.92 EZTaxTaxInclusiveJCode (EZTaxRevJCode deprecated)

Description

EZTaxTaxInclusiveJCode accepts transaction data and performs a tax inclusive calculation to arrive at the base sale amount. The desired total (charge plus all taxes) is placed in the 'charge' field in the transaction, and base charge necessary to generate that total is returned in the 'base_sale' parameter. Taxes for the desired transaction are returned in the same manner as EZTaxJCodeEx does. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using JCodes. The base sale amount will also be stored in the 'optional' field in the log output, converted to an integer.

Format

```
short int EZTaxTaxInclusiveJCode(EZTaxSession session, struct J_CodeEx *trans,  
                                double *base_sale, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[**-1**]: A critical error occurred while preparing the tax inclusive calculation session
[**0**]: No taxes calculated or an error was found
[**n**]: Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session: session handle returned when initialized with call to EZTaxInitEx
*trans: transaction information
*base_sale: calculated charge returned
*err_code: error status returned

Remarks

When both return and err_code are 0, no taxes were found, but there is no error. The base_sale returned will equal the charge sent in. If the charge sent in is insufficient to cover the taxes generated, then no taxes will be calculated and the err_code will be set to -37. An error will be logged to the status file indicating an insufficient charge error occurred.

****Note**:** Non-billable taxes are not included in the calculation of the base sale amount. Adding the base sale amount and all taxes that appear in the log for a transaction may result in a value greater than the target amount, as any non-billable taxes returned by the transaction ***will*** be included in that total.

7.8.93 EZTaxTaxInclusiveNPAN (EZTaxRevNPAN deprecated)

Description

EZTaxTaxInclusiveNPAN accepts transaction data and performs a tax inclusive calculation to arrive at the base sale amount. The desired total (charge plus all taxes) is placed in the 'charge' field in the transaction, and base charge necessary to generate that total is returned in the 'base_sale' parameter. Taxes for the desired transaction are returned in the same manner as EZTaxNPANEx does. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using NPANXXs. The base sale amount will also be stored in the 'optional' field in the log output, converted to an integer.

Format

```
short int EZTaxTaxInclusiveNPAN(EZTaxSession session, struct NPANXX_CodeEx *trans,  
                                double *base_sale, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[**-1**] : A critical error occurred while preparing the tax inclusive calculation session
[**0**] : No taxes calculated or an error was found
[**n**] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session: session handle returned when initialized with call to EZTaxInitEx
*trans: transaction information
*base_sale: calculated charge returned
*err_code: error status returned

Remarks

When both return and err_code are 0, no taxes were found, but there is no error. The base_sale returned will equal the charge sent in. If the charge sent in is insufficient to cover the taxes generated, then no taxes will be calculated and the err_code will be set to -37. An error will be logged to the status file indicating an insufficient charge error occurred.

****Note**:** Non-billable taxes are not included in the calculation of the base sale amount. Adding the base sale amount and all taxes that appear in the log for a transaction may result in a value greater than the target amount, as any non-billable taxes returned by the transaction ***will*** be included in that total.

7.8.94 EZTaxTaxInclusivePCode (EZTaxRevPCode deprecated)

Description

EZTaxTaxInclusivePCode accepts transaction data and performs a tax inclusive calculation to arrive at the base sale amount. The desired total (charge plus all taxes) is placed in the 'charge' field in the transaction, and base charge necessary to generate that total is returned in the 'base_sale' parameter. Taxes for the desired transaction are returned in the same manner as EZTaxPCodeEx does. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. BTN, origination and termination information is passed using PCodes. The base sale amount will also be stored in the 'optional' field in the log output, converted to an integer.

Format

```
short int EZTaxTaxInclusivePCode(EZTaxSession session, struct P_CodeEx *trans,  
                                double *base_sale, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

[-1] : A critical error occurred while preparing the tax inclusive calculation session
[0] : No taxes calculated or an error was found
[n] : Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session: session handle returned when initialized with call to EZTaxInitEx
*trans: transaction information
*base_sale: calculated charge returned
*err_code: error status returned

Remarks

When both return and err_code are 0, no taxes were found, but there is no error. The base_sale returned will equal the charge sent in. If the charge sent in is insufficient to cover the taxes generated, then no taxes will be calculated and the err_code will be set to -37. An error will be logged to the status file indicating an insufficient charge error occurred.

****Note**:** Non-billable taxes are not included in the calculation of the base sale amount. Adding the base sale amount and all taxes that appear in the log for a transaction may result in a value greater than the target amount, as any non-billable taxes returned by the transaction ***will*** be included in that total.

7.8.95 EZTaxTaxInclusiveZip (EZTaxRevZip deprecated)

Description

EZTaxTaxInclusiveZip accepts transaction data and performs a tax inclusive calculation to arrive at the base sale amount. The desired total (charge plus all taxes) is placed in the 'charge' field in the transaction, and base charge necessary to generate that total is returned in the 'base_sale' parameter. Taxes for the desired transaction are returned in the same manner as EZTaxJCodeEx does. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports. The base sale amount will also be stored in the 'optional' field in the log output, converted to an integer.

When the user knows the tax jurisdiction, address information can be supplied to specify it. This function passes tax jurisdiction information to AFC by specifying the appropriate zip code, state identifier, county, and locality. AFC uses this information to determine the JCode used for tax calculations. It is important to supply complete address information as there are many duplicate zip codes and localities contained within the AFC database tables. AFC always returns the first match of the data supplied.

Supplying incomplete information can result in tax calculations for an unexpected jurisdiction. When information regarding the locality is not supplied, AFC may be limited to data based upon the county or parish involved. Likewise, when information about the county is not supplied, the AFC engine may be limited to performance of calculations based upon the state and zip code identifiers.

Format

```
short int EZTaxTaxInclusiveZip(EZTaxSession session, struct zip_codeEx *trans,  
                               double *base_sale, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

- [**-1**]: A critical error occurred while preparing the tax inclusive calculation session
- [**0**]: No taxes calculated or an error was found
- [**n**]: Indicates number of taxes stored in the struct enhanced_taxes_tbl array for this transaction

Parameters

session: session handle returned when initialized with call to EZTaxInitEx
*trans: transaction information
*base_sale: calculated charge returned
*err_code: error status returned

Remarks

When both return and err_code are 0, no taxes were found, but there is no error. The base_sale returned will equal the charge sent in. If the charge sent in is insufficient to cover the taxes generated, then no taxes will be calculated and the err_code will be set to -37. An error will be logged to the status file indicating an insufficient charge error occurred.

*****Note**: Non-billable taxes are not included in the calculation of the base sale amount. Adding the base sale amount and all taxes that appear in the log for a transaction may result in a value greater than the target amount, as any non-billable taxes returned by the transaction *will* be included in that total.***

7.8.96 EZTaxSessionDbVersion

Description

Function EZTaxSessionDbVersion reads the database version being used by the specified AFC Session and returns it to the caller.

Format

```
short int EZTaxDbVersion(EZTaxSession session, char *psz_db_version);
```

Return Value List

TRUE if database version is returned in psz_db_version

FALSE if database version couldn't be obtained

Parameters

session : session handle

*psz_db_version : address of character array that will receive the database version.

Remarks

The database version will be returned in the form ww.xx.yy.zz, where each piece of the database version number can be 1 or 2 digits.

7.8.97 EZTaxSetInvoiceModeEx

Description

Function EZTaxSetInvoiceModeEx sets Invoice Mode on or off - allocating or freeing memory for tables.

This function sets the Invoice Mode on or off. When the Invoice Mode is off AFC works in the normal fashion, each transaction passed is taxed with caps and limits calculated on a per transaction basis. When Invoice Mode is on, AFC performs appropriate tax calculations and returns the taxes calculated, AFC also keeps the tax amounts in memory and uses them to adjust rates for bracketed or tiered taxes and apply caps and limits to the taxes generated for each subsequent transaction. When EZTaxNextCustomerEx is called a summarized version of the taxes is returned.

Format

```
struct enhanced_cust_taxes_tbl *EZTaxSetInvoiceModeEx(EZTaxSession session,  
short int mode);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

pointer : to the tax table if successful
null : if an error occurred

Parameters

session : session handle returned for session when initialized with call to EZTaxInitEx
mode: flag for turning Invoice Mode on (TRUE) or off (FALSE)

References:

[EZTaxNextCustomerEx](#)

Remarks

Interface data structures are also described in detail in the AFC Data Structures section.

Before setting the Invoice Mode to false, EZTaxNextCustomerEx must be called in order to get summarized taxes should be processed prior to turning off the Invoice Mode). Failure to call EZTaxNextCustomerEx will cause the last customer's summarized taxes to be lost.

Note: The summarized taxes returned when calling EZTaxNextCustomerEx are identical to the total of the individual taxes returned from the taxing APIs. Typically one or the other is used but not both. Combining the individual taxes and the summarized taxes will result in twice as much taxes as should be normal.

7.8.98 EZTaxSetInvoiceModeV98

Description

Function EZTaxSetInvoiceModeV98 sets Invoice Mode on or off – allocating or freeing memory for tables. It differs from EZTaxSetInvoiceModeEx in that the tax table returned contains tax category information.

This function sets the Invoice Mode on or off. When the Invoice Mode is off AFC works in the normal fashion, each transaction passed is taxed with caps and limits calculated on a per transaction basis. When Invoice Mode is on, AFC performs appropriate tax calculations and returns the taxes calculated, AFC also keeps the tax amounts in memory and uses them to adjust rates for bracketed or tiered taxes and apply caps and limits to the taxes generated for each subsequent transaction. When EZTaxNextCustomerEx is called a summarized version of the taxes is returned.

Format

```
struct cust_taxes_tbl_v98 *EZTaxSetInvoiceModeV98(EZTaxSession session,  
                                                short int mode);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

pointer : to the tax table if successful
null : if an error occurred

Parameters

session : session handle returned for session when initialized with call to EZTaxInitEx
mode: flag for turning Invoice Mode on (TRUE) or off (FALSE)

References:

EZTaxNextCustomerEx

Remarks

Interface data structures are also described in detail in the AFC Data Structures section.

Before setting the Invoice Mode to false, EZTaxNextCustomerEx must be called in order to get summarized taxes should be processed prior to turning off the Invoice Mode). Failure to call EZTaxNextCustomerEx will cause the last customer's summarized taxes to be lost.

Note: The summarized taxes returned when calling EZTaxNextCustomerEx are identical to the total of the individual taxes returned from the taxing APIs. Typically one or the other is used but not both. Combining the individual taxes and the summarized taxes will result in twice as much taxes as should be normal.

7.8.99 EZTaxSetNexus

Description

Function EZTaxSetNexus sets nexus on or off for the specified states.

This function sets the nexus information by jurisdiction on or off. For each nexus point in the `nexus_table`, the engine will calculate its jurisdiction and either set nexus on or off based on the `has_nexus` variable of the `nexus_table` structure.

Format

```
short int EZTaxSetNexus(EZTaxSession session, struct nexus_table *nexus_tbl,  
                        short int nexus_count);
```

Header(s) Required

`EZTaxProto.h`
`EZTaxDefine.h`
`EZTaxStruct.h`

Libraries

`EZTax2.lib`

Return Value List

TRUE if nexus information was changed;
FALSE otherwise

Parameters

`session` : session handle returned for session when initialized with call to `EZTaxInitEx`
`*nexus_table` : Pointer to nexus table structure that contains nexus information. See the `nexus_table` struct in the header file `EZTaxStruct.h`.
`nexus_count` : Number of nexus points in the structure.

Remarks

Interface data structures are also described in detail in the AFC Data Structures section.

Using this API with the `nexus=on` option in the configuration file is not recommended.
This API is to be used as a programmatic option instead of a configuration file option.

7.8.100 EZTaxSetStateExclusion

Description

EZTaxSetStateExclusion uses the country code and, optionally, a state code to set an exclusion. A flag is used to determine if the country/state jurisdiction is to be turned off (enable regular taxation) or on (exclude country and/or state taxation for all jurisdictions within the country or state including county and local taxation). Transactions for an excluded state continue to return federal (country) level taxes for all transactions within the excluded state.

Note: The three-character country code should be used for countries and the two-character code should be used for states, including US territories.

In this context, references to state apply to either a state or province.

For US territories, the USA Federal exclusions have precedence over US Territory jurisdictions. As a result, all US Territory exclusions will be set or cleared along with US Federal exclusions when only USA is specified.

Please make note, country codes for US territories have been deprecated.

Format

```
short int EZTaxSetStateExclusion(EZTaxSession session, char country_ISO[4],  
                                char state_abv[3], short int flag);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

TRUE : EZTaxSetStateExclusion was successful.
FALSE : EZTaxSetStateExclusion failed.

Parameters

Session : session handle returned when initialized with call to EZTaxInitEx
country_ISO: The country code. These are listed in EZTaxDefine.h.
state_abv: The state abbreviation. These are listed in EZTaxDefine.h.
flag: TRUE - Exclude this state from taxation.
FALSE - Do not exclude this state from taxation.

Remarks

None

[7.8.101 EZTaxSetStateNexus](#)

Description

EZTaxSetStateNexus uses a state code to set the nexus. A flag is used to determine if the state has nexus (true) or not (false).

Format

```
short int EZTaxSetStateNexus(EZTaxSession session, char state_abv[3], short int flag);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

TRUE : EZTaxSetStateNexus was successful.

FALSE : EZTaxSetStateNexus failed.

Parameters

Session : session handle returned when initialized with call to EZTaxInitEx

state_abv: The state abbreviation. These are listed in EZTaxDefine.h.

flag: TRUE or FALSE.

Remarks

None

[7.8.102 EZTaxSetWorkingDir](#)

Description

Function EZTaxSetWorkingDir is utilized to set the working directory to a known location. This is especially useful when AFC is being called from inside a database application where the default working directory may be different from run to run.

Format

```
short int EZTaxSetWorkingDir(char *directory);
```

Header(s) Required

EZTaxProto.h
EZTaxStruct.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

TRUE if change directory is successful;
FALSE otherwise

Parameters

directory : string containing directory path to change to

Remarks

MUST be called before EZTaxInitEx!

[7.8.103 EZTaxThisJCodeEx](#)

Description

EZTaxThisJCodeEx accepts transaction data and performs appropriate tax calculations. The tax jurisdiction to use for tax calculations is specified to AFC with this method. If the AFC Session was started with the tax log open, appropriate taxes are logged and will be reflected in tax reports. Jurisdiction information is passed using a PCode.

Format

```
short int EZTaxThisJCodeEx(EZTaxSession session, struct this_J_Code *trans,  
                           int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

FALSE : Indicates no tax data available for transaction (not taxable) or an error has occurred.
[n] : Indicates number of taxes stored in the struct taxes_tbl array for this transaction

Parameters

session : Session object returned for session when initialized with call to EZTaxInitEx.
*trans : Pointer to data structure that contains transaction data. See the this_J_Code struct in EZTaxStruct.h
*err_code : Error codes returned. See the header EZTaxDefine.h for definitions.

Remarks

This API call is particularly suited for processing local transactions, since there is only one jurisdiction involved

7.8.104 EZTaxThisPCodeEx

Description

EZTaxThisPCodeEx accepts transaction data and performs appropriate tax calculations. The tax jurisdiction to use for tax calculations is specified to AFC with this method. If the AFC Session was started with the tax log open, appropriate taxes are logged and will be reflected in tax reports. Jurisdiction information is passed using a PCode.

Format

```
short int EZTaxThisPCodeEx(EZTaxSession session, struct this_P_Code  
                           *trans, int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

FALSE : Indicates no tax data available for transaction (not taxable) or an error has occurred.
[n]: Indicates number of taxes stored in the struct taxes_tbl array for this transaction.

Parameters

session : Session object returned for session when initialized with call to EZTaxInitEx.
*trans : Pointer to data structure that contains transaction data. See the this_P_Code struct in EZTaxStruct.h
*err_code : Error codes returned. See the header EZTaxDefine.h for definitions.

Remarks

This API call is particularly suited for processing local transactions, since there is only one jurisdiction involved

7.8.105 EZTaxTPPEx

Description

EZTaxTPPEx takes up to three addresses; Ship From, Ship To, and Point-Of-Acceptance. It requires Ship From. If others have NOT NULL values, then they must be valid. Uses rules to determine interstate or intrastate to determine which taxing jurisdiction should be used.

It accepts the optional Nexus Table, which if given will be checked against the jurisdiction determined above. No taxes will be calculated unless the flag is set (by the caller) for the given state. It returns tax_count, having populated the tax_log structure with any taxes found.

Format

```
short int EZTaxTPPEx(EZTaxSession session, struct TPP_addrEx *tpp_trans,  
                      struct nexus_table *nex_tab, short int nexus_count,  
                      int *err_code)
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

Parameters

session : Session object returned for session when initialized with call to EZTaxInitEx.
*tpp_trans : Pointer to data structure that contains transaction data.
*nex_tab : list of states where a business nexus exists.
nexus_count : number of entries in nex_tab
*err_code : Error codes returned. See the header EZTaxDefine.h for definitions.

Remarks

None

7.8.106 EZTaxWriteToLogEx

Description

Function EZTaxWriteToLogEx allows customers to create the correctly formatted binary AFC log from scratch by passing in data they have stored from a different database or saved from AFC transactions using EZTaxInitExp.

Format

```
short int EZTaxWriteToLogEx(EZTaxSession session, int count,  
                           struct EZTax_logEx *log);
```

Header(s) Required

EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

FALSE : An error occurred
TRUE : Success

Parameters

session : Session handle returned when initialized with call to EZTaxInitEx
count: Number of taxes in the log structure
*log : Log structure to save information from

Remarks

EZTaxWriteToLogEx does not support logging of the transaction type and service type.

7.8.107 EZTaxWriteToLogV914

Description

Function EZTaxWriteToLogV914 allows customers to create the correctly formatted binary AFC log from scratch by passing in data they have stored from a different database or saved from AFC transactions.

Format

```
short int EZTaxWriteToLogV914(EZTaxSession session, int count, struct eztax_log_v914 *log
```

Return Value List

FALSE : An error occurred

TRUE : Success

Parameters

session: session handle returned with call to initialize the AFC Session.

count: Number of taxes in the log structure

*log : Log structure containing records to be written to file.

Remarks

EZTaxWriteToLogEx does not support logging of the transaction charge, the tax description, the tax category id or the tax category description.

7.8.108 EZTaxZipEx

Description

EZTaxZipEx accepts transaction data and performs appropriate tax calculations. If the AFC Session was started with the tax log open, tax calculations are logged and will be reflected in tax reports.

When the user knows the tax jurisdiction, address information can be supplied to specify it. This function passes tax jurisdiction information to AFC by specifying the appropriate zip code, state identifier, county, and locality. AFC uses this information to determine the JCode used for tax calculations. It is important to supply complete address information as there are many duplicate zip codes and localities contained within the AFC database tables. AFC always returns the first match of the data supplied.

Supplying incomplete information can result in tax calculations for an unexpected jurisdiction. When information regarding the locality is not supplied, AFC may be limited to data based upon the county or parish involved. Likewise, when information about the county is not supplied, the AFC engine may be limited to performance of calculations based upon the state and zip code identifiers.

Format

```
short int EZTaxZipEx(EZTaxSession session, struct zip_codeEx *trans,  
                      int *err_code);
```

Header(s) Required

EZTaxDefine.h
EZTaxProto.h
EZTaxStruct.h

Libraries

EZTax2.lib

Return Value List

FALSE : Indicates no tax data available for transaction (not taxable) or an error has occurred.
[n] : Indicates number of taxes stored in the struct taxes_tbl array for this transaction.

Parameters

session : Session object returned for session when initialized with call to EZTaxInitEx.
*trans : Pointer to data structure that contains transaction data. See the zip_codeEx struct in the header file EZTaxStruct.h.
*err_code : Error status returned

Remarks

Interface data structures are also described in detail in the AFC Data Structures section. The section AFC Calculations explains how taxes are returned to the user in detail.

AFC currently has Zip Code information for U.S. and Canadian jurisdictions.

The six character Canadian postal code must be broken up for entry. The first 3 characters go into the zip code field and the last 3 characters go into the plus 4 field.

[7.8.109 EZTaxZtoJCodeEx](#)

Description

EZTaxZtoJCodeEx utilizes zip code plus 4 and address information supplied by the user, to obtain a J-Code.

Format

```
unsigned long int EZTaxZtoJCodeEx(EZTaxSession session,  
                                  struct zip_address_p4 *address,  
                                  int *err_code);
```

Header(s) Required

EZTaxProto.h
EZTaxDefine.h

Libraries

EZTax2.lib

Return Value List

[0] : Error Code has been set. If the supplied address was not found or an error has occurred 0 is returned, which is a valid U.S. Federal level code. Please check the error code before using the JCode that is returned.

JCode : Error Code has not been set.

Parameters

session : Session handle returned when initialized with call to EZTaxInitEx
address : Address information with zip+4
*err_code : Error status returned

Remarks

AFC currently has Zip Code information for U.S. and Canadian jurisdictions. The six character Canadian postal code must be broken up for entry. The first 3 characters go into the zip code field and the last 3 characters go into the plus 4 field.

7.8.110 EZTaxZtoPCodeEx

Description

EZTaxZtoPCodeEx returns PCode cross-referenced to a zip code and address.

Format

```
unsigned long int EZTaxZtoPCodeEx(EZTaxSession session struct  
zip_address_p4 *address, int *err_code);
```

Header(s) Required

EZTaxProto.h

Libraries

EZTax2.lib

Return Value List

PCode : Permanent jurisdiction code. If the supplied address was not found or an error has occurred 0 is returned which is a valid U.S. Federal level code. Please check the error code before using the PCode that is returned.

Parameters

session : Session object returned for session when initialized with call to EZTaxInitEx.
*address : Pointer to zip_address_p4 structure containing valid address data, see struct in EZTaxStruct.h Interface data structures are also described in detail in the AFC Data Structures section.
*err_code : Error codes returned. See the header EZTaxDefine.h for definitions.

Remarks

AFC currently has Zip Code information for U.S. and Canadian jurisdictions. The six character Canadian postal code must be broken up for entry. The first 3 characters go into the zip code field and the last 3 characters go into the plus 4 field.

[7.8.111 EZTaxClearSafeHarborOverride](#)

Description

EZTaxClearSafeHarborOverride clears the override flag and the engine will return to using the default safe harbor percentages for the chosen TAMTypeID (celluar type, VoIP type or paging type).

Format

```
EZTaxClearSafeHarborOverride(EZTaxSession session, short TAMTypeID, int * error_code);
```

Return Value List

FALSE: 0

TRUE: 1

Parameters

session: Session object returned for session when initialized with call to EZTaxInitEx.

TAMTypeID

*err_code: Error codes returned. See the header EZTaxDefine.h for definitions.

Remarks

None

[7.8.112 EZTaxSetSafeHarborTAMOverride](#)

Description

EZTaxSetSafeHarborTAMOverride stores the provided fraction in the session object and sets a flag indicating that the particular session has an override at the federal level. The corresponding state level fraction is calculated and stored. The override is written to the EZTax status file if it changed. The AFC engine then applies the new values instead of the default safe harbor percentages for the chosen TAMTypeID.

Format

```
EZTaxSetSafeHarborTAMOverride(EZTaxSession session, short TAMTypeID, double  
originalFederalTAM, double newFederalTAM, int * error_code);
```

Return Value List

FALSE: 0

TRUE: 1

Parameters

session : Session object returned for session when initialized with call to EZTaxInitEx.

originalFederalTAM

newFederalTAM

*err_code: Error codes returned. See the header EZTaxDefine.h for definitions.

Remarks

None

8. Appendices A - G

The complete content of the structures, constants and prototypes are provided in the following Appendices. Also included are the monthly update procedures and a help guide.

- 8.1 EZTaxStruct.h
- 8.2 EZTaxDefine.h
- 8.3 EZTaxProto.h
- 8.4 EZTaxSauStruct.h
- 8.5 EZTaxSauDefine.h
- 8.6 EZTaxSauProto.h
- 8.7 Monthly Update Procedure
- 8.8 Help Guide

8.1 Appendix A EZTaxStruct.h

```
/*
EZTaxStruct.h

The file defines structures that are required to interface with
the AFC system.

*/
#ifndef _inc_EZTaxStruct_
#define _inc_EZTaxStruct_

/* If defined for C++ environment */
#ifndef __cplusplus
extern "C" {
#endif

#include "EZTaxDefine.h"

/* jurisdiction structure returned by EZTaxGetRates */
struct jurisdictionTaxes
{
    unsigned long int      pCode;          /* jurisdiction's PCode */
    short int              taxesCount;     /* count of all taxes for jurisdiction */
    struct enhancedOverride *taxesTable;   /* table of all taxes for jurisdiction in override
format */
    /* notes: (1) the scope value will be set to the tax level */
    /*         (2) the taxesTable should be freed when the results */
    /*             are no longer needed to prevent a memory leak */
};

/* Primary override structure for AFC release 9 */
struct enhancedOverride
{
    short int              scope;          /* Scope of override */
    short int              type;           /* tax type identifier */
    short int              level;          /* tax level identifier */
    short int              dateCount;     /* number of date records (normally 2) */
    struct enhancedDateOverride *dateTable; /* address of array of date records */
};

/* entries in the dateTable for overrides if the tax type is telecom (non-sales tax) */
struct enhancedDateOverride
{
    unsigned long int overrideDate; /* starting (effective) date for this set of tax rates */
    short int        rateCount;    /* number of rate records for this date (normally 1) */
    short int        levelExempt;  /* TRUE indicates tax can be exempted by an */
                                /* exemption for all taxes at the same level as */
                                /* this tax, FALSE indicates it cannot be exempted. */
                                /* NOTE: Tax can be exempted by specific tax exemption. */
    struct enhancedRateOverride *rateTable; /* address of array of telecom rate records */
};

/* rate entries in rateTable for telecom taxes */
struct enhancedRateOverride
{
    double      tax;            /* tax amount (rate) */
    double      max_base;       /* max amount subject to this tax(end of bracket) */
    short int   repl_st;        /* if TRUE tax replaces state tax */
    double      st_ovrd_tax;   /* overrides state rate if present */
    short int   repl_co;        /* if TRUE tax replaces county tax */
    double      co_ovrd_tax;   /* if present, override county tax */
};

/* Deprecated tax override structure (Prior to version 9) */
struct tax_ovrd
{
    short int      scope;        /* Scope of override */
}
```

```

short int          type;           /* tax type identifier */
short int          level;          /* tax level identifier */
short int          level_exempt;   /* TRUE indicates tax can be exempted by an */
                                 /* exemption for all taxes at the same level as */
                                 /* this tax, FALSE indicates it cannot be exempted. */
                                 /* NOTE: Tax can be exempted by specific tax exemption. */
short int          limit;          /* limit - applies to local taxes to indicate */
                                 /* max lines to apply tax to. */
unsigned long int date;           /* tax effective date */
float              tax;            /* tax amount */
float              prev_tax;       /* previous tax amount */
float              max_base;       /* max amount to which tax is */
                                 /* applied - amounts above this */
                                 /* will be taxed at excess rate if */
                                 /* county has excess rate */
float              excess_tax;     /* rate for amount above max_base */
                                 /* returned as part of county tax */
float              st_ovrd_tax;    /* overrides state rate if present */
float              co_ovrd_tax;    /* if present, override county tax */
short int          repl_st;        /* if TRUE tax replaces state tax */
short int          repl_co;        /* if TRUE tax replaces county tax */
};

/* logic Override structure */
struct logicOverride
{
    short int      trans_type;     /* See the programmer user's manual or EZTaxDefine.h */
    short int      srv_type;       /* See the programmer user's manual for valid svc */
    short int      type;          /* tax type identifier */
    short int      level;          /* tax level identifier */
    short int      scope;          /* Scope of override */
    short int      dateCount;     /* number of date records (normally 1) */
    void          *detailTable;    /* pointer to table of detail logic overrides */
};

/* logic Override Detail per effective date */
struct logicOverrideDetail
{
    unsigned long int date;         /* logic effective date */
    short int      calc_type_id;    /* calculation rule */
    short int      data_type_id;    /* tax data type (rate, flat, et cetera */
    short int      eff_sales_id;    /* effect on sales tax (city, co, st) */
    short int      surcharge;       /* indicates that tax is surcharge */
    short int      sale;           /* TRUE, FALSE */
    short int      resale;          /* TRUE, FALSE */
    short int      business;        /* TRUE, FALSE */
    short int      residential;     /* TRUE, FALSE */
    short int      regulated;       /* TRUE, FALSE */
    short int      unregulated;     /* TRUE, FALSE */
    short int      senior_citizen;  /* TRUE, FALSE */
    short int      industrial;     /* TRUE, FALSE */
    short int      lifeline;        /* TRUE, FALSE */
    short int      iloc;           /* TRUE, FALSE */
    short int      clec;           /* TRUE, FALSE */
    short int      primary_ld;      /* TRUE, FALSE */
    short int      primary_local;   /* TRUE, FALSE */
    short int      franchise;       /* TRUE, FALSE */
    short int      nonfranchise;    /* TRUE, FALSE */
    short int      facilities;      /* TRUE, FALSE */
    short int      nonfacilities;   /* TRUE, FALSE */
    short int      tier_at_transaction; /* TRUE, FALSE */
    short int      billable;        /* TRUE, FALSE */
    short int      pro Rated;      /* TRUE, FALSE */
    short int      unrolled_id;     /* 1=Tax Level, 2=City, 3=Local */
    short int      tier_on_total;   /* TRUE, FALSE */
    short int      tax_by_oth;      /* TRUE, FALSE */
    short int      compliance;      /* TRUE, FALSE */
    double          tam;            /* Transaction amount multiplier */
    double          tas;            /* Transaction exempt amount */
    short int      ex_flag;         /* flag showing whether 0-rate */
                                 /* brackets are exempt or not */
};

```

```

short int          taxTaxesCount;           /* count of tax taxes entries */
struct taxTaxesOverride *taxTaxesTable;    /* ptr to table of taxes that this is taxed by */

};

/* logic Override Detail per effective date, with prepaid flag */
struct logicOverrideDetailP
{
    unsigned long int date;                  /* logic effective date */
    short int calc_type_id;                 /* calculation rule */
    short int data_type_id;                 /* tax data type (rate, flat, et cetera */
    short int eff_sales_id;                /* effect on sales tax (city, co, st) */
    short int surcharge;                   /* indicates that tax is surcharge */
    short int sale;                       /* TRUE, FALSE */
    short int resale;                     /* TRUE, FALSE */
    short int business;                   /* TRUE, FALSE */
    short int residential;                /* TRUE, FALSE */
    short int regulated;                  /* TRUE, FALSE */
    short int unregulated;                /* TRUE, FALSE */
    short int senior_citizen;             /* TRUE, FALSE */
    short int industrial;                 /* TRUE, FALSE */
    short int lifeline;                  /* TRUE, FALSE */
    short int lifeline_only;              /* TRUE, FALSE */
    short int ilec;                      /* TRUE, FALSE */
    short int clec;                      /* TRUE, FALSE */
    short int primary_ld;                 /* TRUE, FALSE */
    short int primary_local;              /* TRUE, FALSE */
    short int franchise;                 /* TRUE, FALSE */
    short int nonfranchise;              /* TRUE, FALSE */
    short int facilities;                /* TRUE, FALSE */
    short int nonfacilities;             /* TRUE, FALSE */
    short int tier_at_transaction;       /* TRUE, FALSE */
    short int billable;                  /* TRUE, FALSE */
    short int pro_rated;                 /* TRUE, FALSE */
    short int unrolled_id;                /* 1=Tax Level, 2=City, 3=Local */
    short int tier_on_total;              /* TRUE, FALSE */
    short int tax_by_oth;                 /* TRUE, FALSE */
    short int compliance;                /* TRUE, FALSE */
    double tam;                        /* Transaction amount multiplier */
    double tas;                        /* Transaction exempt amount */
    short int ex_flag;                  /* flag showing whether 0-rate */
                                    /* brackets are exempt or not */
    short int prepaid;                  /* TRUE, FALSE */
    short int taxTaxesCount;            /* count of tax taxes entries */
    struct taxTaxesOverride *taxTaxesTable; /* ptr to table of taxes that this is taxed by */

};

/* logic Override Detail per effective date, with prepaid flag */
struct logicOverrideDetailUseTax
{
    unsigned long int date;                  /* logic effective date */
    short int calc_type_id;                 /* calculation rule */
    short int data_type_id;                 /* tax data type (rate, flat, et cetera */
    short int eff_sales_id;                /* effect on sales tax (city, co, st) */
    short int surcharge;                   /* indicates that tax is surcharge */
    short int sale;                       /* TRUE, FALSE */
    short int resale;                     /* TRUE, FALSE */
    short int business;                   /* TRUE, FALSE */
    short int residential;                /* TRUE, FALSE */
    short int regulated;                  /* TRUE, FALSE */
    short int unregulated;                /* TRUE, FALSE */
    short int senior_citizen;             /* TRUE, FALSE */
    short int industrial;                 /* TRUE, FALSE */
    short int lifeline;                  /* TRUE, FALSE */
    short int lifeline_only;              /* TRUE, FALSE */
    short int ilec;                      /* TRUE, FALSE */
    short int clec;                      /* TRUE, FALSE */
    short int primary_ld;                 /* TRUE, FALSE */
    short int primary_local;              /* TRUE, FALSE */
    short int franchise;                 /* TRUE, FALSE */

```

```

short int      nonfranchise;           /* TRUE, FALSE */
short int      facilities;            /* TRUE, FALSE */
short int      nonfacilities;         /* TRUE, FALSE */
short int      tier_at_transaction;   /* TRUE, FALSE */
short int      billable;              /* TRUE, FALSE */
short int      pro_rated;             /* TRUE, FALSE */
short int      unrolled_id;           /* 1=Tax Level(no Unroll), 2=Cty, 3=Local */
short int      tier_on_total;          /* TRUE, FALSE */
short int      tax_by_oth;             /* TRUE, FALSE */
short int      compliance;            /* TRUE, FALSE */
double         tam;                  /* Transaction amount multiplier */
double         tas;                  /* Transaction exempt amount */
short int      ex_flag;               /* flag showing whether 0-rate */
                                      /* brackets are exempt or not */
short int      sales_flag;            /* TRUE, FALSE */
short int      use_flag;              /* TRUE, FALSE */
short int      prepaid;               /* TRUE, FALSE */
short int      threshold;             /* TRUE, FALSE */
short int      consumed;              /* TRUE, FALSE */
short int      vendor_use;            /* TRUE, FALSE */
short int      taxTaxesCount;          /* count of tax taxes entries */
struct taxTaxesOverride *taxTaxesTable; /* pointer to table of taxes that this is taxed by
*/
};

/* structure for each tax that this tax is taxed by */
struct taxTaxesOverride
{
    short int      tax_type_id;        /* Identifier of tax that has this tax applied to it. */
    short int      tax_level_id;       /* Identifier of level of tax_type_id */
};

/* Individual tax exempt structure */
struct tax_exempt
{
    short int      tax_type;           /* tax type identifier */
    short int      tax_level;          /* tax level identifier */
    unsigned long int exempt_J_Code;  /* for local exemption */
};

/* EZTax_data structure contains data required for tax calculations */
struct EZTax_data
{
    short int      business;           /* TRUE = business customer, FALSE = residential */
    short int      sale;                /* TRUE = Sale, FALSE = Resale */
    short int      regulated;           /* TRUE = Regulated, FALSE = unregulated */
    short int      trans_type;          /* See the programmer user's manual or EZTaxDefine.h */
    short int      srv_type;            /* See the programmer user's manual for valid trans */
                                      /* transaction and service type combinations. */
                                      /* EZTaxDefine.h defines valid service types */
    unsigned long int date;              /* Transaction bill date. Field is provided to */
                                      /* allow rating and taxing to occur on */
                                      /* date other than billing date. */
    float          charge;              /* amount charge to customer for transaction */
    float          minutes;             /* Minutes of call, defaults to zero when */
                                      /* not appropriate (NOTE: some taxes are per minute. */
    int            lines;                /* number of lines (use with transaction type */
                                      /* LOCAL and service type LINES) */
    int            locations;            /* number of locations (use with transaction */
                                      /* type LOCAL and service type LOCATION) */
    short int      incorp;               /* TRUE indicates within incorporated area */
                                      /* of local jurisdiction, FALSE is outside */
    short int      FED_exempt;           /* If TRUE, transaction exempt from Federal Tax */
    short int      st_exempt;             /* If TRUE, transaction exempt from State Tax */
    short int      co_exempt;             /* If TRUE, transaction exempt from County Tax */
    short int      loc_exempt;            /* If TRUE, transaction exempt from local tax */
    unsigned long int FED_J_Code;        /* Jurisdiction for Federal exemption */
    unsigned long int st_J_Code;          /* Jurisdiction for state exemption */
    unsigned long int co_J_Code;          /* Jurisdiction for county exemption */
};

```

```

unsigned long int loc_J_Code; /* Jurisdiction for local exemption */
short int spc_exempt; /* 0 indicates no of special exempts, */
/* other value indicates number of special exempts */
struct tax_exempt *s_exempt; /* Pointer to tax exempt structure that contains the */
/* number of special tax exemptions (spc_exempt) */
unsigned long int inv_no; /* Invoice number, user defined */
unsigned long int srv_lvl_no; /* Service level number, user defined */
unsigned long int optional; /* User defined value for reporting */
char cust_no[CUST_NO_SIZE]; /* Customer number, user defined */
};

/* EZTax_dataEx structure contains data required for tax calulations */
struct EZTax_dataEx
{
    short int customer_type; /* 0=residential, 1=business, 2=Senior Citizen,
3=Industrial */

    short int sale; /* TRUE = Sale, FALSE = Resale */
    short int regulated; /* TRUE = Regulated, FALSE = unregulated */
    short int trans_type; /* See the programmer user's manual or EZTaxDefine.h */
    short int srv_type; /* See the programmer user's manual for valid trans */
    /* transaction and service type combinations. */
    /* EZTaxDefine.h defines valid service types */

    short int business_class; /* CLEC = 1, ILEC = 0 */
    short int service_class; /* Primary Long Distance = 1, Local Service = 0 */
    /* (Default Long Distance) */
    short int facilities_based; /* Facilities Based=TRUE, Non-Facilities Based=FALSE */
    /* (Default Non-Facilities) */
    short int franchise; /* Franchise = TRUE, Non-Franchise = FALSE */
    /* (Default Non-Franchise) */
    short int lifeline; /* Lifeline = TRUE, Non-lifeline = FALSE */
    unsigned long int date; /* Transaction bill date. Field is provided to */
    /* allow rating and taxing to occur on */
    /* date other than billing date. */
    double charge; /* amount charge to customer for transaction */
    double minutes; /* Minutes of call, defaults to zero when not */
    /* appropriate (some taxes are per minute) */
    int lines; /* number of lines (use with transaction type */
    /* LOCAL and service type LINES) */
    int locations; /* number of locations (use with transaction */
    /* type LOCAL and service type LOCATION) */
    short int incorp; /* TRUE indicates within incorporated area */
    /* of local jurisdiction, FALSE is outside */
    short int FED_exempt; /* If TRUE, transaction exempt from Federal Tax */
    short int st_exempt; /* If TRUE, transaction exempt from State Tax */
    short int co_exempt; /* If TRUE, transaction exempt from County Tax */
    short int loc_exempt; /* If TRUE, transaction exempt from local tax */
    unsigned long int FED_J_Code; /* Jurisdiction for Federal exemption */
    unsigned long int st_J_Code; /* Jurisdiction for state exemption */
    unsigned long int co_J_Code; /* Jurisdiction for county exemption */
    unsigned long int loc_J_Code; /* Jurisdiction for local exemption */
    short int spc_exempt; /* 0 indicates no of special exempts, */
    /* other value indicates number of special exempts */
    struct tax_exempt *s_exempt; /* Pointer to tax exempt structure that contains the */
    /* number of special tax exemptions (spc_exempt) */

    short int exempt_type; /* Reason for exemption */
    unsigned long int inv_no; /* Invoice number, user defined */
    unsigned long int srv_lvl_no; /* Service level number, user defined */
    unsigned long int optional; /* User defined value for reporting */
    char cust_no[CUST_NO_SIZE]; /* Customer number, user defined */
    char company_identifier[CO_IDENTIFIER_SIZE]; /* Company Identifier */
    char opt_alpha_1[CUST_NO_SIZE]; /* optional alpha field */
    unsigned long int opt_4; /* optional numeric field */
    unsigned long int opt_5; /* optional numeric field */
    unsigned long int opt_6; /* optional numeric field */
    unsigned long int opt_7; /* optional numeric field */
    unsigned long int opt_8; /* optional numeric field */
    unsigned long int opt_9; /* optional numeric field */
    unsigned long int opt_10; /* optional numeric field */
};

```

```

/* The J_Code structure is used for interfacing with EZTax using J_Codes. */
/* This data structure is passed to EZTax with the function EZTaxJCode. */
struct J_Code
{
    unsigned long int      BTN_J_Code;           /* Bill To Number Jurisdiction Code */
    unsigned long int      orig_J_Code;          /* Origination number Jurisdiction Code */
    unsigned long int      term_J_Code;          /* Termination number Jurisdiction Code */
    struct EZTax_data      tax_data;             /* Required tax information */
};

/* The J_CodeEx structure is used for interfacing with EZTax using J_Codes. */
/* This data structure is passed to EZTax with the extended function EZTaxJCodeEx. */
struct J_CodeEx
{
    unsigned long int      BTN_J_Code;           /* Bill To Number Jurisdiction Code */
    unsigned long int      orig_J_Code;          /* Origination number Jurisdiction Code */
    unsigned long int      term_J_Code;          /* Termination number Jurisdiction Code */
    struct EZTax_dataEx   tax_data;             /* Required tax information */
};

/* The this_J_Code structure is used for interfacing with EZTax using */
/* a specific J_Code. This data structure is passed to EZTax with the */
/* function EZTaxThisJCode. With this function taxes will be calculated */
/* based upon the jurisdiction represented by the specified J_Code. */
struct this_J_Code
{
    unsigned long int      J_Code;                /* Jurisdiction for tax determination */
    struct EZTax_data      tax_data;             /* Required tax information */
};

/* The this_J_CodeEx structure is used for interfacing with EZTax using */
/* a specific J_Code. This data structure is passed to EZTax with the */
/* function EZTaxThisJCodeEx. With this function taxes will be calculated */
/* based upon the jurisdiction represented by the specified J_Code. */
struct this_J_CodeEx
{
    unsigned long int      J_Code;                /* Jurisdiction for tax determination */
    struct EZTax_dataEx   tax_data;             /* Required tax information */
};

/* The this_P_Code structure is used for interfacing with EZTax using */
/* a specific P_Code. This data structure is passed to EZTax with the */
/* function EZTaxThisPCode. With this function taxes will be calculated */
/* based upon the jurisdiction represented by the specified PCode. */
struct this_P_Code
{
    unsigned long int      P_Code;                /* Jurisdiction for tax determination */
    struct EZTax_data      tax_data;             /* Required tax information */
};

/* The this_P_Code structure is used for interfacing with EZTax using */
/* a specific P_Code. This data structure is passed to EZTax with the */
/* function EZTaxThisPCodeEx. With this function taxes will be calculated */
/* based upon the jurisdiction represented by the specified PCode. */
struct this_P_CodeEx
{
    unsigned long int      P_Code;                /* Jurisdiction for tax determination */
    struct EZTax_dataEx   tax_data;             /* Required tax information */
};

/* The priv_line_P_Code structure is used for interfacing with EZTax using */
/* two specifics P_Codes. This data structure is passed to EZTax with the */
/* function EZTaxPrivateLine. With this function taxes will be calculated */
/* based upon the jurisdictions represented by the specified PCode, */
/* allocated to each jurisdiction by the percentage value */
struct private_line_P_Code
{
    unsigned long int      point_A;               /* Jurisdiction for tax determination */
    unsigned long int      point_Z;               /* Jurisdiction for tax determination */
    double                 split;                /* percentage of charge that should */
                                                /* be applied to the first jurisdiction */
};

```

```

    struct EZTax_dataEx tax_data;                                /* Required tax information */
};

/* The P_Code structure is used for interfacing with EZTax using P_Codes. */
/* This data structure is passed to EZTax with the function EZTaxPCode. */
struct P_Code
{
    unsigned long int    BTN_P_Code;                            /* Bill To Number PCode */
    unsigned long int    orig_P_Code;                           /* Origination number PCode */
    unsigned long int    term_P_Code;                           /* Termination number PCode */
    struct EZTax_data   tax_data;                             /* Required tax information */
};

/* The P_Code structure is used for interfacing with EZTax using P_Codes. */
/* This data structure is passed to EZTax with the function EZTaxPCodeEx. */
struct P_CodeEx
{
    unsigned long int    BTN_P_Code;                            /* Bill To Number PCode */
    unsigned long int    orig_P_Code;                           /* Origination number PCode */
    unsigned long int    term_P_Code;                           /* Termination number PCode */
    struct EZTax_dataEx tax_data;                            /* Required tax information */
};

/* The NPANXX structure is used for interfacing with EZTax using NPANXXs. */
/* This data structure is passed to EZTax with the function EZTaxNPAN. */
struct NPANXX_code
{
    unsigned long int    BTN_NPANXX;                           /* Bill To NPANXX Code */
    unsigned long int    orig_NPANXX;                          /* Origination NPANXX Code */
    unsigned long int    term_NPANXX;                          /* Termination NPANXX Code */
    struct EZTax_data   tax_data;                            /* Required tax information */
};

/* The NPANXX_codeEx structure is used for interfacing with EZTax using NPANXXs. */
/* This data structure is passed to EZTax with the function EZTaxNPANEx. */
struct NPANXX_codeEx
{
    unsigned long int    BTN_NPANXX;                           /* Bill To NPANXX Code */
    unsigned long int    orig_NPANXX;                          /* Origination NPANXX Code */
    unsigned long int    term_NPANXX;                          /* Termination NPANXX Code */
    struct EZTax_dataEx tax_data;                           /* Required tax information */
};

/* The zip_address structure is used to pass address data to */
/* EZTax in order to obtain a jurisdiction code. */
struct zip_address
{
    long int             zip_code;                            /* Zip code of taxing jurisdiction */
    unsigned short int   state_id;                            /* state identifier */
    char                county[25];                           /* County name in null terminated string */
    char                locality[26];                          /* Locality name in null terminated string */
};

/* zip_address structure to accommodate Zip code Plus 4 */
struct zip_address_p4
{
    char                zip_code[6];                           /* String containing zip code */
    char                zip_p4[5];                            /* String containing zip_p4 */
    char                country_ISO[4];                      /* String containing country ISO code */
    char                state_abv[3];                          /* String containing state level abbreviation */
    char                county[25];                           /* County name in null terminated string */
    char                locality[26];                          /* Locality name in null terminated string */
};

/* The zip_code structure is used for interfacing with EZTax using address */
/* information. This data structure is passed to EZTax with the function */
/* EZTaxZip. */
struct zip_code
{
    struct zip_address zip_addr;                            /* ZIP code and address information */

```

```

    struct EZTax_data      tax_data;      /* Required tax information */
};

/* Zip code structure for use with Zip plus 4 */
struct zip_code_p4
{
    struct zip_address_p4  zip_addr;     /* ZIP code+4 and address information */
    struct EZTax_data      tax_data;     /* Required tax information */
};

/* Zip code structure for use with Zip plus 4 */
struct zip_codeEx
{
    struct zip_address_p4  zip_addr;     /* ZIP code+4 and address information */
    struct EZTax_dataEx    tax_data;     /* Required tax information */
};

/* The BridgeConferenceTransaction structure is used for calculating tax for a */
/* bridge conference with 1 to n participants. */
/* Note: USA PCode of 0 is considered invalid for Bridge/Billing/Host - for USA must be state
level or lower */
struct BridgeConferenceTransaction
{
    unsigned short int ProcessInvalidParticipant; /* Process invalid Participants w/greatest tax
liability (default=false) */
    unsigned short int IsAdjustment;             /* Flag indicating call is an adjustment or
credit calculation */
    unsigned short int DiscountType;            /* Discount type (for adjustment calculation -
else ignored) */

    int NumberOfParticipants;                   /* Size of participant list */
    unsigned long int *ParticipantPCodeList;    /* Participant PCode List */
    unsigned long int *ParticipantNpaNxxList;   /* Participant NpaNxx List */
    struct zip_address_p4 *ParticipantAddressList; /* Participant Address List */

    unsigned long int BridgePCode;              /* Bridge PCode */
    unsigned long int BridgeNpaNxx;             /* Bridge NpaNxx */
    struct zip_address_p4 BridgeAddress;         /* Bridge Address */

    unsigned long int HostPCode;                /* Host PCode */
    unsigned long int HostNpaNxx;               /* Host NpaNxx */
    struct zip_address_p4 HostAddress;           /* Host Address */

    unsigned long int BillingPCode;              /* Billing PCode */
    unsigned long int BillingNpaNxx;             /* Billing NpaNxx */
    struct zip_address_p4 BillingAddress;         /* Billing Address */

    struct EZTax_dataEx trans_data;             /* Required transaction information */
};

struct BridgeConferenceParticipant
{
    int ParticipantId;                         /* By order received - 1 to N */
    int Offset;                               /* Where in tax table participant starts */
    int TaxCount;                             /* Number of taxes in tax table for participant */
    int ErrorCode;                            /* Error code (if applicable) for participant */
    short int TransactionType;                /* The transaction type used for participant */
    short int ServiceType;                   /* The service type used for participant */
};

struct BridgeConferenceTaxes
{
    struct EZTax_log_v914 *BCTaxes; /* Individual Participant Taxes for Conference */
    unsigned short int TaxCount; /* Total number of taxes in BCTaxes array */
    struct BridgeConferenceParticipant* BCParticipants; /* Participant array */
    int ParticipantCount; /* Number of Participants in BCParticipants array */
};

```

```

/* Structures use by EZTaxTPP and EZTaxAdjTPP */
struct TPP_addr
{
    struct zip_address_p4    ship_from;      /* origin ZIP code+4 and address information */
    struct zip_address_p4    ship_to;        /* delivery ZIP code+4 and address information */
    struct zip_address_p4    point_of_accpt; /* bill-to ZIP code+4 and address information */
    struct EZTax_data        tax_data;       /* Required tax information */
};

/* Structures use by EZTaxTPPEX and EZTaxAdjTPPEX */
struct TPP_addrEx
{
    struct zip_address_p4    ship_from;      /* origin ZIP code+4 and address information */
    struct zip_address_p4    ship_to;        /* delivery ZIP code+4 and address information */
    struct zip_address_p4    point_of_accpt; /* bill-to ZIP code+4 and address information */
    struct EZTax_dataEx     tax_data;       /* Required tax information */
};

struct nexus_table
{
    char          state[3];                /* state abbreviation */
    short int     have_nexus;             /* TRUE, FALSE */
};

/* The taxes structure defines the data structure used to return tax
 * information to the user. The functions EZTaxJCode, EZTaxNPAN, and
 * EZTaxZip each return a short int. Zero indicates that no taxes were
 * calculated for the transaction passed. If taxes were calculated, the
 * value returned indicates the number of taxes returned in an array
 * of struct taxes with the size of array being the number of taxes,
 * Use the following definition to access the calculated taxes */
/* "extern struct taxes *tax_tbl;". The following code segment
 * demonstrates an acceptable method of retrieving calculated tax data.
*/
/*
/*   struct taxes_tbl      tax;
/*   extern struct taxes_tbl  *EZTax_table;
/*
/*       for (i=0; i<rtn_value; i++);
/*
/*       {
/*           tax = EZTax_table[i];
/*
/*           -- Perform billing system required task
/*
/*       }
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
// The taxes structure defines the data structure used to return tax
 * information to the user. The functions EZTaxJCode, EZTaxNPAN, and
 * EZTaxZip each return a short int. Zero indicates that no taxes were
 * calculated for the transaction passed. If taxes were calculated, the
 * value returned indicates the number of taxes returned in an array
 * of struct taxes with the size of array being the number of taxes,
 * Use the following definition to access the calculated taxes */
/* "extern struct taxes *tax_tbl;". The following code segment
 * demonstrates an acceptable method of retrieving calculated tax data.
*/
/*
/*   struct taxes_tbl      tax;
/*   extern struct taxes_tbl  *EZTax_table;
/*
/*       for (i=0; i<rtn_value; i++);
/*
/*       {
/*           tax = EZTax_table[i];
/*
/*           -- Perform billing system required task
/*
/*       }
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
// The enhanced_taxes_tbl structure provides for additional precision */
struct enhanced_taxes_tbl
{
    unsigned long int    p_code;      /* p-code for tax */
    short int            tax_type;    /* Tax type */
    short int            tax_level;   /* Tax level */

```

```

short int          calc_type;      /* Calc type; RATE, FIXED, PER_MINUTE, PER_LINE,
SELFTAXING_RATE */
double            rate;           /* Tax rate or amount applied */
double            tax_amount;     /* Calculated tax amount */
double            taxable_measure; /* Amount of charge + any taxed taxes */
double            exempt_sale_amt; /* Amount of the charge exempt from taxes */
char              *desc;          /* Tax description string */
short int         billable;        /* Billable flag from tax logic */
short int         compliance;     /* Compliance flag from tax logic */
short int         surcharge_flag; /* Surcharge flag from tax logic */
short int         TVcalc_type_id; /* for EZView */
short int         TVsurcharge;    /* for EZView */
short int         TVtax_type_id[EZVIEW_ID_ARRAY]; /* for EZView */
short int         TVtax_level_id[EZVIEW_ID_ARRAY]; /* for EZView */
};

/* The taxes_tbl_v98 structure adds tax category information */
struct taxes_tbl_v98
{
    unsigned long int p_code;        /* p-code for tax */
    short int        tax_type;       /* Tax type */
    short int        tax_level;      /* Tax level */
    short int        calc_type;     /* Calc type; RATE, FIXED, PER_MINUTE, PER_LINE,
SELFTAXING_RATE */
    double           rate;           /* Tax rate or amount applied */
    double           tax_amount;     /* Calculated tax amount */
    double           taxable_measure; /* Amount of charge + any taxed taxes */
    double           exempt_sale_amt; /* Amount of the charge exempt from taxes */
    char             *desc;          /* Tax description string */
    short int         billable;        /* Billable flag from tax logic */
    short int         compliance;     /* Compliance flag from tax logic */
    short int         surcharge_flag; /* Surcharge flag from tax logic */
    short int         tax_cat_id;     /* Tax category */
    char             *tax_cat_desc;   /* Tax category description string */
    reserved[RESERVE_SIZE]; /* reserved for BillSoft use */
    TVcalc_type_id; /* for EZView */
    TVsurcharge;    /* for EZView */
    TVtax_type_id[EZVIEW_ID_ARRAY]; /* for EZView */
    TVtax_level_id[EZVIEW_ID_ARRAY]; /* for EZView */
};

/* The customer taxes table is similar to the taxes_tbl structure. See comments above */
/* This table is returned to the user from EZTaxSetCustMode(). This table contains a summarized */
*/
/* tax information per customer */
struct cust_taxes_tbl
{
    unsigned long int j_code;        /* Jurisdiction code for tax */
    short int        tax_type;       /* Tax type */
    short int        tax_level;      /* Tax level */
    short int        calc_type;     /* Calc type; RATE, FIXED, PER_MINUTE, PER_LINE,
SELFTAXING_RATE */
    float            rate;           /* Tax rate or amount applied */
    double           tax_amount;     /* Calculated tax amount */
    char             *desc;          /* Tax description string */
    TVsurcharge;    /* for EZView */
    int              lines;          /* Number of lines from customer input */
    int              locations;     /* Number of locations from customer input */
    float            max_base;       /* max amount to which tax is applied */
    /* amounts above this will be taxed at excess rate if */
    /* county has excess rate */
    float            excess_tax;     /* rate for amount above max base */
    /* returned as part of county tax */
    int              limit;          /* limit on the amount a charge can be taxed*/
    double           total_charge;   /* Sum of charges calc. on per customer basis */
};

/* The enhanced invoice(customer) taxes table is similar to the taxes_tbl and */
/* cust_taxes_tbl structures. See comments for these two other structures */

```

```

/* above. This table is returned to the user from EZTaxSetCustModeEx() or
EZTaxSetInvoiceModeEx(). */
/* This table contains a summarized tax information per customer or invoice */
#define enhanced_invoice_taxes_tbl enhanced_cust_taxes_tbl
struct enhanced_cust_taxes_tbl
{
    unsigned long int      j_code;          /* j-code for tax */
    short int              tax_type;        /* Tax type */
    short int              tax_level;       /* Tax level */
    short int              calc_type;       /* Calculation type (RATE, FIXED, etc.) */
    double                 rate;            /* Tax rate or amount applied */
    double                 tax_amount;      /* Calculated tax amount */
    double                 exempt_sale_amt; /* Amount of the charge exempt from taxes */
    char                  *desc;           /* Tax description string */
    short int              TVsurcharge;     /* for EZView */
    int                   lines;           /* # of lines from customer input */
    int                   locations;       /* # of locations from customer input */
    double                minutes;         /* # of minutes from customer input */
    double                max_base;        /* max amount to which tax is applied. */
                                         /* Amounts above this will be taxed at a */
                                         /* higher bracketed rate (if applicable). */
    double                min_base;        /* min amount to which tax is applied */
    double                excess_tax;      /* rate for amount above max_base returned */
                                         /* as part of county tax */
    double                total_charge;    /* Sum of charges calc. on per customer basis */
    void                 *cust_tax_data;   /* Customer mode tax data for logging */
};

/* The invoice(customer) taxes table V98 is similar to the taxes_tbl and */
/* cust_taxes_tbl structures. See comments for these two other structures */
/* above. This table is returned to the user from EZTaxSetCustModeV98() or
EZTaxSetInvoiceModeV98(). */
/* This table contains a summarized tax information per customer or invoice */
#define invoice_taxes_tbl_v98 cust_taxes_tbl_v98
struct cust_taxes_tbl_v98
{
    unsigned long int      j_code;          /* j-code for tax */
    short int              tax_type;        /* Tax type */
    short int              tax_level;       /* Tax level */
    short int              calc_type;       /* Calculation type (RATE, FIXED, etc.) */
    double                 rate;            /* Tax rate or amount applied */
    double                 tax_amount;      /* Calculated tax amount */
    double                 exempt_sale_amt; /* Amount of the charge exempt from taxes */
    char                  *desc;           /* Tax description string */
    short int              TVsurcharge;     /* for EZView */
    int                   lines;           /* # of lines from customer input */
    int                   locations;       /* # of locations from customer input */
    double                minutes;         /* # of minutes from customer input */
    double                max_base;        /* max amount to which tax is applied. */
                                         /* Amounts above this will be taxed at a */
                                         /* higher bracketed rate (if applicable). */
    double                min_base;        /* min amount to which tax is applied */
    double                excess_tax;      /* rate for amount above max_base returned */
                                         /* as part of county tax */
    double                total_charge;    /* Sum of charges calc. on per customer basis */
    short int              tax_cat_id;     /* Tax category */
    char                  *tax_cat_desc;   /* Tax category description string */
};

/* declare address structure */
struct address_data
{
    short int              tax_level;       /* Federal, State, County/Parish, Local */
    char                  locality[26];    /* City name string */
    char                  county[25];      /* County name string */
    char                  state[2];        /* Two character state abbreviation */
    long int               zip_begin;       /* Lowest zip code in range for identified area */
    long int               zip_end;         /* Highest zip code in range for identified area */
};

```

```

struct address_data_p4
{
    short int tax_level; /* Tax level identifier */
    char locality[26]; /* City name string */
    char county[25]; /* County name string */
    char state[2]; /* Two character state abbreviation */
    char country_ISO[4]; /* Country ISO code */
    char zip_begin[6]; /* zip code or beginning of range */
    char zipb_p4[5]; /* Zip begin plus 4 */
    char zip_end[6]; /* zip code or end of range */
    char zipe_p4[5]; /* Zip end plus 4 */
};

struct file_path
{
    char *EZTax_data; /* EZTax db file */
    char *EZTax_IDX; /* EZTax idx file */
    char *EZTax_DLL; /* EZTax dll */
    char *EZTax_log; /* EZTax log */
    char *EZTax_npanxx; /* EZTax NPANXX file */
    char *EZTax_status; /* EZTax status file */
    char *EZTax_temp_file; /* EZTax temporary file */
    char *EZTax_location; /* EZTax location description file */
    char *EZTax_zip; /* EZTax zip code file */
    char *EZTax_customer_key; /* EZTax customer key file */
    char *EZTax_pcode; /* EZTax PCODE file */
    char *EZTax_jcode; /* EZTax JCODE file */
    char *EZTax_over; /* EZTax override file */
};

struct file_path_v914
{
    char *EZTax_data_dir; /* EZTax data directory */
    char *EZTax_work_dir; /* Working directory */
    char *EZTax_log_file; /* EZTax log */
    char *EZTax_status_file; /* EZTax status file */
    char *EZTax_ovr_file; /* EZTax override file */
    char *EZTax_exc_file; /* EZTax exclusion file */
    char *EZTax_nex_file; /* EZTax nexus file */
    char *EZTax_bdl_file; /* EZTax bundle file */
};

/* structure used when doing local logging using EZTaxInitExp */
struct taxes_tbl_exp
{
    struct taxes_tbl *taxtable; /* pointer to tax table */
    struct EZTax_log *EZTaxlog; /* pointer to EZTax binary log */
};

/* Tax log structure, Use this structure for writing to EZTax binary log. */
struct EZTax_log
{
    unsigned long int p_code; /* from tax table, convert jcode to pcode */
    short int tax_type; /* from tax table */
    short int tax_level; /* from tax table */
    double tax; /* from tax table */
    double tax_amount; /* from tax table */
    double sale_amt; /* taxable measure from tax table */
    double exempt_sale_amt; /* taxable measure if tax = 0 */
    double refund_uncollect; /* taxable measure if tax < 0 */
    double net_taxable_sale; /* reserved by EZTax, default to 0 */
    double minutes; /* customer input */
    unsigned long int inv_no; /* optional, user defined invoice number */
    unsigned long int srv_lvl_no; /* optional, service type */
    unsigned long int optional; /* optional, transaction type */
    int lines; /* customer input */
    int locations; /* customer input */
    short int calc_type; /* calculation type from tax table */
    char cust_no[CUST_NO_SIZE]; /* optional, user defined customer identifier */
    int reserved1; /* reserved by EZTax, default to 0 */
}

```

```

int             reserved2;          /* reserved by EZTax, default to 0 */
double          reserved3;          /* reserved by EZTax, default to 0 */
};

/* Tax log structure, use this structure with EZTaxWriteToLogEx for writing to EZTax log. */
struct EZTax_logEx
{
    unsigned long int p_code;           /* from tax table, convert jcode to pcode */
    int             tax_type;          /* from tax table */
    short int       tax_level;         /* from tax table */
    double          tax;               /* from tax table */
    double          tax_amount;        /* from tax table */
    double          sale_amt;          /* taxable measure from tax table */
    double          exempt_sale_amt;   /* taxable measure if tax = 0 */
    double          refund_uncollect; /* taxable measure if tax < 0 */
    double          minutes;          /* customer input */
    unsigned long int inv_no;          /* optional, user defined invoice number */
    unsigned long int srv_lvl_no;     /* optional, service type */
    unsigned long int optional;        /* optional, transaction type */
    int             lines;             /* customer input */
    int             locations;         /* customer input */
    short int       calc_type;         /* calculation type from tax table */
    short int       billable;          /* billable flag from tax_grp */
    short int       compliance;        /* compliance flag from tax_grp */
    char            cust_no[CUST_NO_SIZE]; /* optional, user defined customer identifier */
    int             adj_type;          /* adjustment type from customer input */
    int             exempt_type;       /* exemption type from customer input */
    int             surcharge_flag;    /* surcharge flag from tax table */
    unsigned long int log_counter;     /* counter of log records */
    char            company_identifier[CO_IDENTIFIER_SIZE]; /* customer number */
    char            opt_alpha_1[CUST_NO_SIZE]; /* optional user input */

    unsigned long int opt_4;           /* optional user input */
    unsigned long int opt_5;           /* optional user input */
    unsigned long int opt_6;           /* optional user input */
    unsigned long int opt_7;           /* optional user input */
    unsigned long int opt_8;           /* optional user input */
    unsigned long int opt_9;           /* optional user input */
    unsigned long int opt_10;          /* optional user input */
};

struct EZTax_log_v914
{
    /* EZTax_logEx entries */
    unsigned long int p_code;           /* from tax table, convert jcode to pcode */
    int             tax_type;          /* from tax table */
    short int       tax_level;         /* from tax table */
    double          tax;               /* from tax table */
    double          tax_amount;        /* from tax table */
    double          sale_amt;          /* taxable measure from tax table */
    double          exempt_sale_amt;   /* taxable measure if tax = 0 */
    double          refund_uncollect; /* taxable measure if tax < 0 */
    double          minutes;          /* customer input */
    unsigned long int inv_no;          /* optional, user defined invoice number */
    unsigned long int srv_lvl_no;     /* optional, service type */
    unsigned long int optional;        /* optional, transaction type */
    int             lines;             /* customer input */
    int             locations;         /* customer input */
    short int       calc_type;         /* calculation type from tax table */
    short int       billable;          /* billable flag from tax_grp */
    short int       compliance;        /* compliance flag from tax_grp */
    char            cust_no[CUST_NO_SIZE]; /* optional, user defined customer identifier */
    int             adj_type;          /* adjustment type from customer input */
    int             exempt_type;       /* exemption type from customer input */
    int             surcharge_flag;    /* surcharge flag from tax table */
    unsigned long int log_counter;     /* counter of log records */
    char            company_identifier[CO_IDENTIFIER_SIZE]; /* customer number */
    char            opt_alpha_1[CUST_NO_SIZE]; /* optional user input */

    unsigned long int opt_4;           /* optional user input */
    unsigned long int opt_5;           /* optional user input */
    unsigned long int opt_6;           /* optional user input */
};

```

```

        unsigned long int    opt_7;           /* optional user input */
        unsigned long int    opt_8;           /* optional user input */
        unsigned long int    opt_9;           /* optional user input */
        unsigned long int    opt_10;          /* optional user input */

/* Extended entries */
short int          trans_type;          /* transaction type */
short int          srv_type;           /* service type */
double             trans_charge;        /* applied trans charge amount */
char               *desc;              /* Tax description string */
short int          tax_cat_id;         /* Tax category */
char               *tax_cat_desc;       /* Tax category description string */

void*              int_data;            /* pointer to internal data */

};

/* Reconcile transaction report data struct - extended tsr report data fields */
struct rtr_data
{
    int reportId;          /* eg: Report_Canada or 1 */
    char location[32];     /* Holds text for country or state name */
    short int stateId;     /* State ID */
    short int ctryId;      /* Country ID */
    int bundlePkg;         /* eg: 20000 (maps to bundle transaction type) */
    int bundleId;          /* eg: 20008 (maps to bundle service type) */
    int transType;         /* Transaction type */
    int svcType;           /* Service Type */
    double charge;          /* Client supplied charge */
    double calcCharge;     /* Calculated charge (for tax inclusive calcs) */
};

/* no-tax transaction structure, use with EZTaxGetNoTaxTrans */
struct no_tax_tbl
{
    unsigned long int    p_code;          /* jurisdiction applied */
    short int            trans_type;      /* transaction type from input transaction */
    short int            srv_type;       /* service type from input transaction */
    short int            attr;           /* attribute from input transaction */
    double               charge;          /* charge amount from input transaction */
    company_identifier[CO_IDENTIFIER_SIZE]; /* cust num from input trans */
    cust_no[CUST_NO_SIZE];   /* customer number from input transaction */
};

#endif __cplusplus
#endif
#endif /* _inc_EZTaxStruct_ */

```

8.2 Appendix B EZTaxDefine.h

```
EZTaxDefine.h

*/
#ifndef _inc_EZTaxDefine_
#define _inc_EZTaxDefine_

/* if defined for C++ environment */
#ifndef __cplusplus
extern "C" {
#endif

/* add a couple of platform-independence defines */
#if defined(WIN32)
    #undef PF_V8_TAXABLE_MEASUR

/* function export define */
    #ifndef PF_FUNC_SPEC
        #ifdef BSI_EXPORT_DLL
            #define PF_FUNC_SPEC __declspec (dllexport)
        #else
            #define PF_FUNC_SPEC __declspec (dllimport)
        #endif
    #endif
#else
    #define PF_V8_TAXABLE_MEASURE 1
#endif

/* function export define */
#define PF_FUNC_SPEC
#endif

/* Define CACHE_ALL - Used to request caching of entire file */
#define CACHE_ALL 1000000

#define EZTAX_VERSION "9.19.1801.1"
#define EZTAX_MAJOR_VERSION 9
#define EZTAX_DB_VERSION 19

/* Define VERSION_MAX_LENGTH: Maximum length of EZTAX_VERSION plus '\0' */
/* MINOR_VERSION is build number (eg: 1403) */
/* INCR_VERSION is build iteration (0-9) - the limit of 9 is from assembly version limits */
#define VERSION_MAX_LENGTH 13
#define MINOR_VERSION_MIN 0
#define MINOR_VERSION_MAX 9999
#define INCR_VERSION_MIN 0
#define INCR_VERSION_MAX 9

/* define maximum length for a starting directory path */
#define MAX_PATH_LEN 256

/* define upper limit for max_base in tax tables */
#define MAX_BASE_LIMIT 2147483647

typedef void *EZTaxSession;

/* Define TRUE and FALSE */
#ifndef TRUE
#define TRUE 1
#endif

#ifndef FALSE
#define FALSE 0
#endif

/* Define IO status values */
#ifndef MORE
#define MORE 0
#endif
```

```

#endif

#ifndef FOUND
#define FOUND
#endif

#ifndef LAST
#define LAST
#endif

#ifndef NO_IO_YET
#define NO_IO_YET
#endif

#ifndef NOT_FOUND
#define NOT_FOUND
#endif

/* Define size of Cust_no in tax_log */
#define CUST_NO_SIZE 20

/* Define size of tax type description here so users */
/* will know the max size to store the description */
#define TAX_DESCRIPTION_LENGTH 51

/* Define size of company_identifier */
#define CO_IDENTIFIER_SIZE 20

/* Define size of EZView arrays inside taxes table */
#define EZVIEW_ID_ARRAY 10

/* define size of reserved space in taxes table */
#define RESERVE_SIZE 20

/* define adjustment method types */
#define ADJUSTMENT_DEFAULT 0
#define ADJUSTMENT_LEAST_FAVORABLE 1
#define ADJUSTMENT_MOST_FAVORABLE 2
#define ADJUSTMENT_NOT_APPLICABLE 3

/* Define default regulated/unregulated setting */
#define DEFAULT_REGULATED_UNREGULATED FALSE

/* Define customer types */
#define RESIDENTIAL 0
#define BUSINESS 1
#define SENIOR_CITIZEN 2
#define INDUSTRIAL 3
#define DEFAULT_CUSTOMER_TYPE 1

/* Define business classes */
#define CLEC 1
#define ILEC 0
#define DEFAULT_BUSINESS_CLASS 1

/* Define service classes */
#define PRIMARY_LONG_DISTANCE 1
#define PRIMARY_LOCAL_SERVICE 0
#define DEFAULT_SERVICE_CLASS 1

/* Define sale/resale/consumed values */
#define WHOLESALE 0 /* Resale and Wholesale are synonymous */
#define RESALE 0
#define SALE_RETAIL 1 /* Sale and Retail are synonymous */
#define SALE 1
#define CONSUMED 2
#define VENDOR_USE 3

/* Define relationship values */

```

```

#define EQUAL_TO 0
#define LESS_THAN -1
#define GREATER_THAN 1
#define UNINITIALIZED 86
#define SAME_SIGN 2

/* Define Error Codes */
#define INVALID_TRANSACTION -1
#define JCODE_NOT_FOUND -11
#define PCODE_NOT_FOUND -12
#define ZIPCODE_NOT_FOUND -13
#define NPANXX_NOT_FOUND -14
#define ADDRESS_NOT_FOUND -15
#define CO_ST_ZP_NOT_FOUND -16
#define SESSION_NOT_INIT -17
#define JCDB_NOT_OPEN -18
#define PCDB_NOT_OPEN -19
#define ZIPDB_NOT_OPEN -20
#define NPDB_NOT_OPEN -21
#define ADDB_NOT_OPEN -22
#define JURISDICTION_NOT_FOUND -23
#define FIPS_NOT_FOUND -24
#define FIPS_NOT_OPEN -25
#define PCODEFIPS_NOT_OPEN -26
#define INVALID_TRANSACTION_DATE -27
#define INVALID_TRANSRV_PAIR -28
#define LOG_NAME_NOT_SET -29
#define INVALID_ATTR -30
#define INVALID_PROPERTY -31
#define TS_NOT_SUPPORTED -32
#define NEXUS_ERROR -33
#define TS_NOT_LICENSED -34
#define SYSTEM_MEMORY_ERROR -35
#define PRIVATE_LINE_SPLIT_OUT_OF_RANGE -36
#define REV_CALC_SESSION_FAILURE -37
#define TAX_INCLUSIVE_CALC_SESSION_FAILURE -37
#define TAX_ON_TAX_FAILURE -38
#define CUSTOM_LOG_SIZE_EXCEEDED -39
#define INVALID_EXCLUSION -40
#define INVALID_BRIDGECONF_DATA -41
#define INVALID_SAFE_HARBOR_TYPE -42
#define INVALID_SAFE_HARBOR_FRACTION -43
#define BUNDLEDITEM_TAX_ITEMS_EXCEEDED -44
#define BUNDLED_FIXED_EXCEEDS_CHARGE -45
#define BUNDLED_TS_PAIR_NOT_FOUND -46
#define BUNDLED_TS_PAIR_INVALID_SYNTAX -47
#define INTERSTATE_NOT_VALID_FOR_INTERNATIONAL_CALLS -48

```

```

/* Codes set by function EZTaxTPP and EZTaxAdjTPP */
#define MISSING_OR_INVALID_SHIP_FROM -27
#define INVALID_SHIP_TO_ADDRESS -28
#define INVALID_POINT_OF_ACCEPTANCE -29
#define INVALID_STATE_INPUT -30
#define ZIPCODE_DATABASE_NOT_OPEN -31

/* Deprecation error */
#define FUNCTION_DEPRECATED -50

/* Define size of getRates return table for RPG interface */
#define RPG_GETRATES_ARRY_SIZE 200

/* Codes set by RPG Wrapper APIs */
#define GETRATES_TAXES_TABLE_OVERFLOW -200
#define OVERRIDE_TAXES_TABLE_CNT_NOT_ONE_OR_MORE -201
#define OVERRIDE_INPUT_LARGER_THAN_DEFLT_ARRY_SIZE -202
#define OVERRIDE_CONTAINS_DUPLICATE_BRACKET_IN_RATES -203
#define MORE_THAN_ONE_TAX_STRUCT_PASSED_TO_OVERRIDE -204

```

```

/* Define user bundle id start range */
#define BUNDLE_ID_START 20000

/* Group options for tax calculation results */
#define DEFAULT 0
#define GROUP_SAME_LEVEL 1
#define GROUP_CO_LOCAL 3
#define GROUP_ST_CO_LOCAL 4
#define GROUP_SALES 8
#define GROUP_SALES_CATEGORY 24

/* define file indicators */
#define NPANXX 1
#define ZIPCODE 2
#define ADDRESS 3
#define PCODE 4
#define JCODE 5
#define FIPS 6
#define PCDFIPS 7

/* Define tax levels */
#define FEDERAL_LEVEL 0
#define STATE_LEVEL 1
#define COUNTY_LEVEL 2
#define LOCAL_LEVEL 3
#define COUNTY_LEVEL_U 4
#define OTHER_LEVEL 5
#define ST_CO_LOCAL_LEVEL 6
#define CO_LOCAL_LEVEL 7

/* Define tax data calculation types. */
#define RATE 1
#define FIXED 2
#define PER_MINUTE 3
#define PER_LINE 4
#define SELFTAXING_RATE 5
#define PER_BRACKET 6
#define FIXED_ON_TIER 7

/* Customer mode specific calculations */
#define CUST_PER_LINE 8
#define CUST_PER_BRACKET 9
#define CUST_PER_LINE_TOTAL 10
#define CUST_PER_BRACKET_TOTAL 11

/* Define sort filters */
#define SORT_NO_FILTER_SET 0
#define SORT_FILTER_CUST_PER 1
#define SORT_FILTER_CUST_TOT 2

/* Define discount types */
#define NO_DISCOUNT 0
#define RETAIL_PRODUCT 1
#define MANUFACTURER_PRODUCT 2
#define ACCOUNT_LEVEL 3
#define SUBSIDIZED 4
#define GOODWILL 5

/* Define exemptions */
#define NO_EXEMPTION_TYPE 0
#define FED_SALES_SUPREMECY 1
#define STATE_LOCAL_GOV_SALES 2
#define SALES_FOR_RESALE 3
#define FED_COUPONS_WIC_VOUCHERS 4
#define REDUCED_FOOD_RATE 5
#define NON_PROFIT_SALES 6
#define PUBLIC SCHOOL_SALES 7
#define RELIGIOUS_CHARITABLE_SALES 8
#define PRESCRIPTION_DRUG_SALES 9
#define PROSTHO_ORTHO_DEVICES 10
#define INSULIN_SALES 11
#define INTER_STATE_FOREIGN_SALES 12

```

```

#define INGREDIENT_COMPONENT_PARTS 13

/* Define tax-on-tax iterations */
#define TAX_ON_TAX_ONCE 0
#define TAX_ON_TAX_ITERATE_ON_TAX_AMOUNT 1
#define TAX_ON_TAX_ITERATE_ON_TAXABLE_MEASURE 2

/* Define Safe Harbor Override Types */
#define EZTAX_SAFE_HARBOR_CELLULAR_TYPE 1
#define EZTAX_SAFE_HARBOR_VOIP_TYPE 2
#define EZTAX_SAFE_HARBOR_PAGING_TYPE 4
#define EZTAX_SAFE_HARBOR_CLEAR_ALL 7

/* Pro-rated adjustment related defines */
#define NO_PRORATED_APPLICATION 0
#define STANDARD_PRORATED_ADJUSTMENT 1
#define CANCEL_PRORATED_CALCULATION 2

#ifndef __cplusplus
#endif

#endif /* _inc_EZTaxDefine_ */

```

EZTaxTaxType.h

```
#ifndef _inc_EZTaxTaxTypeDefine_
#define _inc_EZTaxTaxTypeDefine_

#ifndef __cplusplus
extern "C" {
#endif

/* Define tax types */
#define NO_TAX 0 /*No Tax*/
#define SALES_TAX 1 /*Sales Tax*/
#define BUSINESS_AND_OCCUPATION_TAX 2 /*Business and Occupation Tax*/
#define CARRIER_GROSS_RECEIPTS 3 /*Carrier Gross Receipts*/
#define DISTRICT_TAX 4 /*District Tax*/
#define EXCISE_TAX 5 /*Excise Tax*/
#define FEDERAL_EXCISE_TAX 6 /*Federal Excise Tax*/
#define FED_USF_A_SCHOOL 7 /*Fed USF A - School*/
#define LICENSE_TAX 8 /*License Tax*/
#define PUC_FEE 9 /*P.U.C. Fee*/
#define E911 10 /*E-911*/
#define SERVICE_TAX 11 /*Service Tax*/
#define SPECIAL_TAX 12 /*Special Tax*/
#define STATE_UNIVERSAL_SERVICE_FUND 13 /*State Universal Service Fund*/
#define STATUTORY_GROSS_RECEIPTS 14 /*Statutory Gross Receipts*/
#define SURCHARGE 15 /*Surcharge*/
#define UTILITY_USERS_TAX 16 /*Utility Users Tax*/
#define SALES_WEB_HOSTING 17 /*Sales (Web Hosting)*/
#define FED_UNIVERSAL_SERVICE_FUND 18 /*Fed Universal Service Fund*/
#define STATE_HIGH_COST_FUND 19 /*State High Cost Fund*/
#define STATE_DEAF_AND_DISABLED_FUND 20 /*State Deaf and Disabled Fund*/
#define CA_TELECONNECT_FUND 21 /*CA Teleconnect Fund*/
#define UNIVERSAL_LIFELINE_TELEPHONE_SERVICE_CHARGE 22 /*Universal Lifeline Telephone Service Charge*/
#define TELECOM_RELAY_SURCHARGE 23 /*Telecom Relay Surcharge*/
#define TELECOMMUNICATIONS_INFRASTRUCTURE_MAINTENANCE_FEE 24 /*Telecommunications Infrastructure Maintenance Fee*/
#define POISON_CONTROL_FUND 25 /*Poison Control Fund*/
#define TELECOMMUNICATIONS_INFRASTRUCTURE_FUND 26 /*Telecommunications Infrastructure Fund*/
#define NY_MCTD_186C 27 /*NY MCTD 186c*/
#define NY_MCTD_184A 28 /*NY MCTD 184a*/
#define FRANCHISE_TAX 29 /*Franchise Tax*/
#define UTILITY_USERS_TAX_BUSINESS 30 /*Utility Users Tax - Business*/
#define FED_TELECOMMUNICATIONS_RELAY_SERVICE 31 /*Fed Telecommunications Relay Service*/
#define DISTRICT_TAX_RESIDENTIAL_ONLY 32 /*District Tax (Residential Only)*/
#define TRANSIT_TAX 33 /*Transit Tax*/
#define TELECOMMUNICATIONS_ASSISTANCE_SERVICE_FUND 34 /*Telecommunications Assistance Service Fund*/
#define E911_BUSINESS 35 /*E911 (Business)*/

#endif
```

```

#define TRS_BUSINESS 36 /*TRS
(Business)*/
#define UNIVERSAL_SERVICE_FUND_LINE 37 /*Universal
Service Fund (Line)*/
#define UNIVERSAL_SERVICE_FUND_BUSINESS_LINE 38 /*Universal
Service Fund (Business Line)*/
#define E911_PBX_TRUNK_LINE 39 /*E911 (PBX/Trunk
line)*/
#define LICENSE_TAX_BUSINESS 40 /*License Tax
(Business)*/
#define OPTIONAL_TIMF 41 /*Optional TIMF*/
#define SALES_TAX_BUSINESS 42 /*Sales Tax
(Business)*/
#define E911_RESIDENTIAL 43 /*E911
(Residential)*/
#define E911_WIRELESS 44 /*E911
(Wireless)*/
#define NY_FRANCHISE_184 45 /*NY Franchise
184*/
#define NY_FRANCHISE_184_USAGE 46 /*NY Franchise
184 Usage*/
#define NY_MCTD_184A_USAGE 47 /*NY MCTD 184a
Usage*/
#define UNIVERSAL_SERVICE_FUND_WIRELESS 48 /*Universal
Service Fund (Wireless)*/
#define USE_TAX 49 /*Use Tax*/
#define SALES_TAX_DATA 50 /*Sales Tax
(Data)*/
#define MUNICIPAL_RIGHT_OF WAY 51 /*Municipal Right
of Way*/
#define MUNICIPAL_RIGHT_OF_WAY_BUSINESS 52 /*Municipal Right
of Way (Business)*/
#define MUNICIPAL_RIGHT_OF_WAY_PRIVATE_LINE 53 /*Municipal Right
of Way (Private Line)*/
#define UTILITY_USERS_TAX_WIRELESS 54 /*Utility Users
Tax - Wireless*/
#define FED_USF_CELLULAR 55 /*Fed USF
Cellular*/
#define FED_USF_PAGING 56 /*Fed USF
Paging*/
#define SALES_TAX_INTERSTATE 57 /*Sales Tax
(Interstate)*/
#define UTILITY_USERS_TAX_PBX_TRUNK 58 /*Utility Users
Tax (PBX Trunk)*/
#define DISTRICT_TAX_WEB_HOSTING 59 /*District Tax
(Web Hosting)*/
#define CA_HIGH_COST_FUND_A 60 /*CA High Cost
Fund A*/
#define TELECOMMUNICATIONS_EDUCATION_ACCESS_FUND 61 /*Telecommunications Education Access Fund*/
#define FED_TRS_CELLULAR 62 /*Fed TRS
Cellular*/
#define FED_TRS_PAGING 63 /*Fed TRS
Paging*/
#define COMMUNICATIONS_SERVICE_TAX 64 /*Communications
Service Tax*/
#define VALUE_ADDED_TAX 65 /*Value Added Tax
(VAT)*/
#define GOODS_AND_SERVICE_TAX 66 /*Goods and
Service Tax (GST)*/
#define HARMONIZED_SALES_TAX 67 /*Harmonized
Sales Tax (HST)*/
#define PROVINCIAL_SALES_TAX 68 /*Provincial
Sales Tax (PST)*/
#define QUEBEC_SALES_TAX 69 /*Quebec Sales
Tax (QST)*/
#define NATIONAL_CONTRIBUTION_REGIME 70 /*National
Contribution Regime (NCR)*/
#define UTILITY_USERS_TAX_CABLE_TELEVISION 71 /*Utility Users
Tax (Cable Television)*/

```

#define FCC_REGULATORY_FEE	72	/*FCC Regulatory Fee*/
#define FRANCHISE_TAX_CABLE(Cable)*/	73	/*Franchise Tax
#define UNIVERSAL_SERVICE_FUND_PAGING Service Fund (Paging)*/	74	/*Universal
#define STATUTORY_GROSS_RECEIPTS_WIRELESS Receipts (Wireless)*/	75	/*Statutory Gross
#define SGT_E911_TAX	76	/*Sage E911 Tax*/
#define SGT_E911_TAX_BUSINESS(Business)*/	77	/*Sage E911 Tax
#define SGT_E911_TAX_PBX_TRUNK_LINE(PBX/Trunk line)*/	78	/*Sage E911 Tax
#define SGT_E911_TAX_RESIDENTIAL(Residential)*/	79	/*Sage E911 Tax
#define SGT_E911_TAX_WIRELESS(Wireless)*/	80	/*Sage E911 Tax
#define SGT_LICENSE_TAX Tax*/	81	/*Sage License
#define FRANCHISE_TAX_WIRELESS(Wireless)*/	82	/*Franchise Tax
#define FEDERAL_USF_ALTERNATE(Alternate)*/	83	/*Federal USF
#define PEG_ACCESS_FEE Education and Government (PEG) Access Fee*/	84	/*Public
#define COMMUNICATIONS_SERVICE_TAX_SATELLITE Service Tax (Satellite)*/	85	/*Communications
#define FRANCHISE_TAX_SATELLITE(Satellite)*/	86	/*Franchise Tax
#define CARRIER_COST_RECOVERY Recovery*/	87	/*Carrier Cost
#define FEDERALTRS_ALTERNATE(Alternate)*/	88	/*Federal TRS
#define TRS_CENTREX	89	/*TRS (Centrex)*/
#define UTILITY_USERS_TAX_CABLE_TELEVISION_BUSINESS Tax (Cable Television - Business)*/	90	/*Utility Users
#define UTILITY_USERS_TAX_CENTREX Tax (Centrex)*/	91	/*Utility Users
#define E911_CENTREX(Centrex)*/	92	/*E911
#define UTILITY_USERS_TAX_LINE Tax (Line)*/	93	/*Utility Users
#define CRIME_CONTROL_DISTRICT_TAX District Tax*/	94	/*Crime Control
#define LIBRARY_DISTRICT_TAX District Tax*/	95	/*Library
#define HOSPITAL_DISTRICT_TAX District Tax*/	96	/*Hospital
#define HEALTH_SERVICES_DISTRICT_TAX District Tax*/	97	/*Health Services
#define EMERGENCY_SERVICES_DISTRICT_TAX Services District Tax*/	98	/*Emergency
#define IMPROVEMENT_DISTRICT_TAX District Tax*/	99	/*Improvement
#define DEVELOPMENT_DISTRICT_TAX District Tax*/	100	/*Development
#define TRANSIT_WEB_HOSTING_TAX Hosting Tax*/	101	/*Transit Web
#define AMBULANCE_DISTRICT_TAX District Tax*/	102	/*Ambulance
#define FIRE_DISTRICT_TAX Tax*/	103	/*Fire District
#define POLICE_DISTRICT_TAX Tax*/	104	/*Police District
#define FOOTBALL_DISTRICT_TAX District Tax*/	105	/*Football
#define BASEBALL_DISTRICT_TAX District Tax*/	106	/*Baseball
#define CRIME_CONTROL_DISTRICT_WEB_HOSTING_TAX District Web Hosting Tax*/	107	/*Crime Control

#define LIBRARY_DISTRICT_WEB_HOSTING_TAX	108	/*Library
District Web Hosting Tax*/		
#define HOSPITAL_DISTRICT_WEB_HOSTING_TAX	109	/*Hospital
District Web Hosting Tax*/		
#define HEALTH_SERVICES_DISTRICT_WEB_HOSTING_TAX	110	/*Health Services
District Web Hosting Tax*/		
#define EMERGENCY_SERVICES_DISTRICT_WEB_HOSTING_TAX	111	/*Emergency
Services District Web Hosting Tax*/		
#define IMPROVEMENT_DISTRICT_WEB_HOSTING_TAX	112	/*Improvement
District Web Hosting Tax*/		
#define DEVELOPMENT_DISTRICT_WEB_HOSTING_TAX	113	/*Development
District Web Hosting Tax*/		
#define UTILITY_USERS_TAX_INTERSTATE	114	/*Utility Users
Tax (Interstate)*/		
#define UTILITY_USERS_TAX_TELEGRAPH	115	/*Utility Users
Tax (Telegraph)*/		
#define E911_NETWORK_AND_DATABASE_SURCHARGE	116	/*E911 Network
And Database Surcharge*/		
#define LICENSE_TAX_EMERGENCY	117	/*License Tax
Emergency*/		
#define LICENSE_TAX_EMERGENCY_BUSINESS	118	/*License Tax
Emergency (Business)*/		
#define EDUCATIONAL_SALES_TAX	119	/*Educational
Sales Tax*/		
#define EDUCATIONAL_USE_TAX	120	/*Educational Use
Tax*/		
#define E911_OPERATIONAL_SURCHARGE_COUNTY_COMMISSION	121	/*E911
Operational Surcharge County Commission*/		
#define E911_OPERATIONAL_SURCHARGE_VOTER_APPROVED	122	/*E911
Operational Surcharge Voter Approved*/		
#define SALES_TAX_NINE_HUNDRED	123	/*Sales Tax Nine
Hundred*/		
#define CONVENTION_CENTER_TAX	124	/*Convention
Center Tax*/		
#define E911_HIGH_CAPACITY_TRUNK	125	/*E911 High
Capacity Trunk*/		
#define SCHOOL_BOARD_TAX_A	126	/*School Board
Tax A*/		
#define SCHOOL_BOARD_TAX_B	127	/*School Board
Tax B*/		
#define SCHOOL_BOARD_TAX_C	128	/*School Board
Tax C*/		
#define SCHOOL_BOARD_TAX_D	129	/*School Board
Tax D*/		
#define SCHOOL_BOARD_TAX_E	130	/*School Board
Tax E*/		
#define SCHOOL_BOARD_TAX_F	131	/*School Board
Tax F*/		
#define SCHOOL_DISTRICT_TAX	132	/*School District
Tax*/		
#define POLICE_JURY_TAX_B	133	/*Police Jury Tax
B*/		
#define POLICE_JURY_TAX_C	134	/*Police Jury Tax
C*/		
#define POLICE_JURY_TAX_E	135	/*Police Jury Tax
E*/		
#define COMMUNICATIONS_SERVICE_TAX_WIRELESS	136	/*Communications
Service Tax (Wireless)*/		
#define SERVICE_PROVIDER_TAX	137	/*Service
Provider Tax*/		
#define TELECOMMUNICATIONS_SALES_TAX	138	
/*Telecommunications Sales Tax*/		
#define ADVANCED_TRANSIT_TAX	139	/*Advanced
Transit Tax*/		
#define ADVANCED_TRANSIT_WEB_HOSTING_TAX	140	/*Advanced
Transit Web Hosting Tax*/		
#define MISSOURI_UNIVERSAL_SERVICE_FUND	141	/*Missouri
Universal Service Fund*/		
#define BUSINESS_AND_OCCUPATION_TAX_WHOLESALE	142	/*Business and
Occupation Tax (Wholesale)*/		

#define TELECOMMUNICATIONS_EDUCATION_ACCESS_FUND_CENTREX	143	/*Telecommunications Education Access Fund (Centrex)*/
#define BUSINESS_AND_OCCUPATION_TAX_OTHER	144	/*Business and Occupation Tax (Other)*/
#define TRIBAL_SALES_TAX	145	Tax*/
#define SALES_TAX_DATA_PROCESSING	146	/*Sales Tax (Data Processing)*/
#define TRANSIT_TAX_DATA_PROCESSING	147	(Data Processing)/*
#define CRIME_CONTROL_DISTRICT_TAX_DATA_PROCESSING	148	District Tax (Data Processing)/*
#define LIBRARY_DISTRICT_TAX_DATA_PROCESSING	149	District Tax (Data Processing)/*
#define HOSPITAL_DISTRICT_TAX_DATA_PROCESSING	150	District Tax (Data Processing)/*
#define HEALTH_SERVICES_DISTRICT_TAX_DATA_PROCESSING	151	District Tax (Data Processing)/*
#define EMERGENCY_SERVICES_DISTRICT_TAX_DATA_PROCESSING	152	Services District Tax (Data Processing)/*
#define IMPROVEMENT_DISTRICT_TAX_DATA_PROCESSING	153	District Tax (Data Processing)/*
#define DEVELOPMENT_DISTRICT_TAX_DATA_PROCESSING	154	District Tax (Data Processing)/*
#define ADVANCED_TRANSIT_TAX_DATA_PROCESSING	155	Transit Tax (Data Processing)/*
#define CA_PSPE_SURCHARGE	156	Surcharge*/
#define DISTRICT_TAX_DATA_PROCESSING	157	(Data Processing)/*
#define TAX_TYPE_RESERVED_158	158	/*Eschelon UUT*/
#define CABLE_FRANCHISE_FEE	159	Fee*/
#define STATUTORY_GROSS_RECEIPTS_BUSINESS	160	Receipts (Business)/*
#define E911_VOIP	161	/*E911 (VoIP)*/
#define FUSF_VOIP	162	/*FUSF (VoIP)*/
#define FUSF	163	/*FUSF*/
#define COST_RECOVERY_SURCHARGE	164	Surcharge*/
#define UNIVERSAL_SERVICE_FUND_VOIP	165	Service Fund (VoIP)/*
#define COMMUNICATIONS_SERVICE_TAX_CABLE	166	Service Tax (Cable)/*
#define MUNICIPAL_RIGHT_OF_WAY_CABLE	167	of Way (Cable)/*
#define TAX_TYPE_RESERVED_168	168	/*Reserved*/
#define FCC_REGULATORY_FEE_WIRELINE	169	Fee (Wireline)/*
#define FCC_REGULATORY_FEE_WIRELESS	170	Fee (Wireless)/*
#define TAX_TYPE_RESERVED_171	171	/*Reserved*/
#define STATUTORY_GROSS_RECEIPTS_VIDEO	172	Receipts (Video)/*
#define UTILITY_USERS_TAX_LIFELINE	173	Tax - Lifeline*/
#define TRS_LONG_DISTANCE	174	Distance*/
#define TELECOM_RELAY_SURCHARGE_WIRELESS	175	Surcharge (Wireless)/*
#define SALES_TAX_SENIOR_CITIZEN	176	Senior Citizen*/
#define REGULATORY_COST_CHARGE_LOCAL	177	Charge - Local*/
#define REGULATORY_COST_CHARGE_INTRASTATE	178	Charge - Intrastate*/
#define REGULATORY_COST_CHARGE_CABLE	179	Charge - Cable*/
#define PUC_FEE_CABLE	180	Cable*/

#define PROVINCIAL_SALES_TAX_TOLL	181	/* Provincial
Sales Tax (TOLL) */		
#define UUT	182	/* UUT */
#define TAX_TYPE_RESERVED_183	183	/* Reserved */
#define SALES_TAX_MANUFACTURING	184	/* Sales Tax-
Manufacturing*/		
#define USE_TAX_MANUFACTURING	185	/* Use Tax-
Manufacturing*/		
#define SALES_TAX_MOTOR_VEHICLES	186	/* Sales Tax-Motor
Vehicles*/		
#define USE_TAX_MOTOR_VEHICLES	187	/* Use Tax-Motor
Vehicles*/		
#define RENTAL_TAX	188	/* Rental Tax */
#define RENTAL_TAX_LINEN	189	/* Rental Tax-
Linen*/		
#define SALES_TAX_VENDING	190	/* Sales Tax-
Vending*/		
#define RENTAL_TAX_MOTOR_VEHICLES	191	/* Rental Tax-
Motor Vehicles*/		
#define SALES_TAX_WHOLESALE	192	/* Sales Tax-
Wholesale*/		
#define SALES_TAX_FOOD_AND_DRUGS	193	/* Sales Tax-Food
and Drugs*/		
#define SALES_TAX_FOOD	194	/* Sales Tax-
Food*/		
#define FUR_TAX	195	/* Fur Tax */
#define PRIVILEGE_TAX_MANUFACTURING	196	/* Privilege Tax-
Manufacturing*/		
#define LEAD_ACID_BATTERY_FEE	197	/* Lead Acid
Battery Fee*/		
#define SALES_TAX_MOTOR_FUEL	198	/* Sales Tax-Motor
Fuel*/		
#define LEAD_ACID_BATTERY_FEE_LARGER_BATTERY	199	/* Lead Acid
Battery Fee-Larger Battery*/		
#define SALES_TAX_PARKING	200	/* Sales Tax-
Parking*/		
#define PRIVILEGE_TAX_RECREATION	201	/* Privilege Tax-
Recreation*/		
#define DRY_CLEANING_FEE	202	/* Dry Cleaning
Fee*/		
#define WHITE_GOODS_TAX	203	/* White Goods
Tax*/		
#define SALES_TAX_MEDICAL_EQUIPMENT	204	/* Sales Tax-
Medical Equipment*/		
#define ELECTRONIC_WASTE_RECYLCLING_FEE_SMALL	205	/* Electronic
Waste Recycling Fee-Small*/		
#define ELECTRONIC_WASTE_RECYLCLING_FEE_MEDIUM	206	/* Electronic
Waste Recycling Fee-Medium*/		
#define ELECTRONIC_WASTE_RECYLCLING_FEE_LARGE	207	/* Electronic
Waste Recycling Fee-Large*/		
#define ALCOHOLIC_BEVERAGE_TAX	208	/* Alcoholic
Beverage Tax*/		
#define SALES_TAX_ALCOHOL	209	/* Sales Tax-
Alcohol*/		
#define LIQUOR_DRINK_TAX	210	/* Liquor Drink
Tax*/		
#define IN_UNIVERSAL_SERVICE_CHARGE	211	/* IN Universal
Service Charge*/		
#define TRS_PAGING	212	/* TRS (Paging) */
#define CONNECTME_FUND	213	/* ConnectME
Fund*/		
#define PA_PURTA_SURCHARGE	214	/* PA PURTA
Surcharge*/		
#define CONNECTME_FUND_VOIP	215	/* ConnectME Fund
(VoIP)*/		
#define CONNECTME_FUND_CABLE	216	/* ConnectME Fund
(Cable)*/		
#define TRS_VOIP	217	/* TRS (VoIP) */
#define CONSUMER_COUNSEL_FEE	218	/* Consumer
Counsel Fee*/		

#define SAN_DIEGO_UNDERGROUND_CONVERSION_SURCHARGE	219	/*San Diego
Underground_Conversion_Surcharge*/		
#define RSPF_SURCHARGE	220	/*RSPF
Surcharge*/		
#define NETWORK_ACCESS_FEE	221	/*Network Access
Fee*/		
#define FRANCHISE_FEE	222	/*Franchise Fee*/
#define CASF	223	/*CASF*/
#define LICENSE_TAX_CABLE	224	/*License Tax
(Cable)*/		
#define RELAY_MISSOURI_SURCHARGE	225	/*Relay Missouri
Surcharge*/		
#define FCC_REGULATORY_FEE_VOIP	226	/*FCC Regulatory
Fee (VoIP)*/		
#define TAX_TYPE_RESERVED_227	227	/*Reserved*/
#define MUNICIPAL_RIGHT_OF_WAY_EXTENSION	228	/*Municipal Right
of Way (Extension)*/		
#define CARRIER_COST_RECOVERY_VOIP	229	/*Carrier Cost
Recovery (VoIP)*/		
#define SALES_TAX_VIDEO	230	/*Sales Tax-
Video*/		
#define NORTH_CAROLINA_TELECOMMUNICATIONS_SALES_TAX	231	/*North Carolina
Telecommunications Sales Tax*/		
#define TELECOMMUNICATIONS_RELAY_SURCHARGE_CELLULAR	232	
/*Telecommunications Relay Surcharge (Cellular)*/		
#define E911_PREPAID_WIRELESS	233	/*E-911 Prepaid
Wireless*/		
#define TELECOMMUNICATIONS_RELAY_SURCHARGE_PAGING	234	
/*Telecommunications Relay Surcharge (Paging)*/		
#define TELECOMMUNICATIONS_RELAY_SURCHARGE_VOIP	235	
/*Telecommunications Relay Surcharge (VoIP)*/		
#define TDAP	236	/*TDAP*/
#define TAP_SURCHARGE	237	/*TAP Surcharge*/
#define COMMUNICATIONS_SERVICE_TAX_NON_FACILITIES	238	/*Communications
Service Tax (Non-Facilities)*/		
#define E911_VOIP_ALTERNATE	239	/*E-911 (VoIP)
Alternate*/		
#define E911_VOIP_PBX	240	/*E-911 (VoIP
PBX)*/		
#define UTILITY_USERS_TAX_VOIP	241	/*Utility Users
Tax (VoIP)*/		
#define UTILITY_USERS_TAX_VOIP_BUSINESS	242	/*Utility Users
Tax (VoIP-Business)*/		
#define SOLID_WASTE_COLLECTION_TAX	243	/*Solid Waste
Collection Tax*/		
#define E911_VOIP_BUSINESS	244	/*E-911 (VoIP
Business)*/		
#define E911_VOIP_NOMADIC	245	/*E-911 (VoIP-
Nomadic)*/		
#define E911_PREPAID_WIRELESS_ALTERNATE	246	/*E-911 Prepaid
Wireless (Alternate)*/		
#define POLICE_AND_FIRE_PROTECTION_FEE	247	/*Police and Fire
Protection Fee*/		
#define SAN_FRANCISCO_ACCESS_LINE_TAX	248	/*San Francisco
Access Line Tax*/		
#define SAN_FRANCISCO_ACCESS_LINE_TAX_PBX_TRUNK_LINE	249	/*San Francisco
Access Line Tax (PBX/Trunk Line)*/		
#define SAN_FRANCISCO_ACCESS_LINE_TAX_VOIP	250	/*San Francisco
Access line Tax (VoIP)*/		
#define SAN_FRANCISCO_ACCESS_LINE_TAX_WIRELESS	251	/*San Francisco
Access Line Tax (Wireless)*/		
#define SAN_FRANCISCO_ACCESS_LINE_TAX_HIGH_CAP_TRUNK	252	/*San Francisco
Access Line Tax (High Cap Trunk)*/		
#define CITY_OF_SAN_JOSE_TELEPHONE_LINE_TAX	253	/*City of San
Jose Telephone Line Tax*/		
#define CITY_OF_SAN_JOSE_TELEPHONE_LINE_TAX_PBX_TRUNK_LINE	254	/*City of San
Jose Telephone Line Tax-PBX/Trunk Line*/		
#define CITY_OF_SAN_JOSE_TELEPHONE_LINE_TAX_VOIP	255	/*City of San
Jose Telephone Line Tax (VoIP)*/		
#define CITY_OF_SAN_JOSE_TELEPHONE_LINE_TAX_WIRELESS	256	/*City of San
Jose Telephone Line Tax (Wireless)*/		

#define SAN_LEANDRO_EMERG_COM_SYS_ACCESS_TAX	257	/*San Leandro
Emerg Com Sys Access Tax*/		
#define SAN_LEANDRO_EMERG_COM_SYS_ACCESS_TAX_PBX_TRUNK	258	/*San Leandro
Emerg Com Sys Access Tax (PBX Trunk)*/		
#define SAN_LEANDRO_EMERG_COM_SYS_ACCESS_TAX_VOIP	259	/*San Leandro
Emerg Com Sys Access Tax (VoIP)*/		
#define SAN_LEANDRO_EMERG_COM_SYS_ACCESS_TAX_WIRELESS	260	/*San Leandro
Emerg Com Sys Access Tax (Wireless)*/		
#define SAN_LEANDRO_EMERG_COM_SYS_ACCESS_TAX_HIGH_CAP_TRNK	261	/*San Leandro
Emerg Com Sys Access Tax-High Cap Trnk*/		
#define POLICE_AND_FIRE_PROTECTION_FEE_PREPAID	262	/*Police and Fire Protection Fee (Prepaid)*/
#define PUBLIC_SAFETY_COMMUNICATIONS_SURCHARGE	263	/*Public Safety Communications Surcharge*/
#define E911_TECHNICAL_CHARGE	264	/*E-911 Technical Charge*/
#define TELECOM_ASSISTANCE_SVC_FUND_HIGH_CAPACITY_TRUNK	265	/*Telecom Assistance Svc Fund-High Capacity Trunk*/
#define CRTLEVY	266	/*CRT Levy*/
#define ACCESS_LINE_TAX	267	/*Access Line Tax*/
Tax*/		
#define ACCESS_LINE_TAX_PBX_TRUNK_LINE	268	/*Access Line Tax (PBX/Trunk Line)*/
#define ACCESS_LINE_TAX_VOIP	269	/*Access Line Tax (VoIP)*/
#define ACCESS_LINE_TAX_WIRELESS	270	/*Access Line Tax (Wireless)*/
#define WI_USF	271	/*WI USF*/
#define NETWORK_ACCESS_FEE_INTERSTATE	272	/*Network Access Fee-Interstate*/
#define SALES_TAX_OTHER	273	/*Sales Tax - Other*/
#define FCC_REGULATORY_FEE_VOIP_ALTERNATE	274	/*FCC Regulatory fee (VoIP Alternate)*/
#define EXCISE_TAX_WIRELESS	275	/*Excise Tax (Wireless)*/
#define RESERVED_276	276	/*Reserved_276*/
#define FEDERAL_UNIVERSAL_SERVICE_FUND_NON_BILLABLE	277	/*Federal Universal Service Fund (Non-Billable)*/
#define MUNICIPAL_RIGHT_OF WAY_HIGH_CAPACITY_TRUNK	278	/*Municipal Right of Way-High Capacity Trunk*/
of Way-High Capacity Trunk*/		
#define EDUCATION_CESS	279	/*Education Cess*/
#define SECONDARY_AND_HIGHER_EDUCATION_CESS	280	/*Secondary and Higher Education Cess*/
#define UTILITY_USERS_TAX_VIDEO	281	/*Utility Users Tax (Video)*/
#define STATE_USF_VOIP_ALTERNATE	282	/*State USF (VoIP Alternate)*/
#define TRS_VOIP_BUSINESS	283	/*TRS (VoIP Business)*/
#define TRS_TRUNK	284	/*TRS (Trunk) */
#define DEAF_AND_DISABLED_FUND_WIRELESS	285	/*Deaf and Disabled Fund (Wireless)*/
Disabled Fund (Wireless)*/		
#define UTILITY_USERS_TAX_WIRELESSBUSINESS	286	/*Utility Users Tax-Wireless(Business)*/
#define TELECOMMUNICATIONS_SALES_TAX_PREPAID	287	/*Telecommunications Sales Tax-Prepaid*/
/*Telecommunications Sales Tax-Prepaid*/		
#define CA_HIGH_COST_FUND_A_VOIP_ACTUAL	288	/*CA High Cost Fund A (VoIP Actual)*/
Fund A (VoIP Actual)*/		
#define STATE_HIGH_COST_FUND_VOIP_ACTUAL	289	/*State High Cost Fund (VoIP Actual)*/
Fund (VoIP Actual)*/		
#define UNIVERSAL_LIFELINE_TELEPHONE_SVC_CHG_VOIP_ACTUAL	290	/*Universal Lifeline Telephone Svc Chg (VoIP Actual)*/
Lifeline Telephone Svc Chg (VoIP Actual)*/		
#define TELECOMMUNICATIONS_RELAY_SVC_CHARGE_VOIP_ACTUAL	291	/*Telecommunications Relay Svc Charge (VoIP Actual)*/
/*Telecommunications Relay Svc Charge (VoIP Actual)*/		
#define CA_TELECONNECT_FUND_VOIP_ACTUAL	292	/*CA Teleconnect Fund (VoIP Actual)*/
Fund (VoIP Actual)*/		
#define CASF_VOIP_ACTUAL	293	/*CASF (VoIP Actual)*/
Actual)*/		

#define OKLAHOMA_SALES_TAX	294	/*Oklahoma Sales Tax*/
#define BUSINESS_AND_OCCUPATION_TAX_PRTG_AND_PUBLISHING	295	/*Business and Occupation Tax (Prtg and Publishing)*/
#define PREMIER_RESORT_AREA_TAX	296	/*Premier Resort Area Tax*/
#define E911_EQUALIZATION_SURCHARGE	297	/*E911 Equalization Surcharge*/
#define UNIVERSAL_SERVICE_FEE	298	/*Universal Service Fee*/
#define NE_UNIVERSAL_SERVICE	299	/*NE Universal Service*/
#define TAP_SURCHARGE_WIRELESS	300	/*TAP Surcharge (Wireless)*/
#define GA_UNIVERSAL_ACCESS_FUND	301	/*GA Universal Access Fund*/
#define CA_HIGH_COST_FUND_A_WIRELESS	302	/*CA High Cost Fund A (Wireless)*/
#define CA_TELECONNECT_FUND_WIRELESS	303	/*CA Teleconnect Fund (Wireless)*/
#define CASF_WIRELESS	304	/*CASF (Wireless)*/
#define STATE_HIGH_COST_FUND_WIRELESS	305	/*State High Cost Fund (Wireless)*/
#define PUC_FEE_WIRELESS	306	/*PUC Fee (Wireless)*/
#define UNIVERSAL_LIFELINE_TELEPHONE_SVC_CHARGE_WIRELESS	307	/*Universal Lifeline Telephone Svc Charge (Wireless)*/
#define NY_TAF	308	/*NY TAF*/
#define PREPAID_WIRELESS_E911_TRS_SURCHARGE	309	/*Prepaid Wireless E911 TRS Surcharge*/
#define TRS_PREPAID_WIRELESS	310	/*TRS-Prepaid Wireless*/
#define FUSF_MULTI_LINE	311	/*FUSF (Multi-line)*/
#define ND_GROSS_RECEIPTS_TAX	312	/*ND Gross Receipts Tax*/
#define NY_SALES_TAX	313	/*NY Sales Tax*/
#define NY_LOCAL_TRANSIT_TAX	314	/*NY Local Transit Tax*/
#define NY_LOCAL_DISTRICT_TAX	315	/*NY Local District Tax*/
#define SALES_TAX_SATELLITE	316	/*Sales Tax-Satellite*/
#define SALES_TAX_COMMERCIALLEASE	317	/*Sales Tax-Commercial Lease*/
#define FOOD_AND_BEVERAGE_TAX	318	/*Food and Beverage Tax*/
#define NETWORK_ACCESS_FEE_LD_INTERSTATE	319	/*Network Access Fee LD-Interstate*/
#define NETWORK_ACCESS_FEE_LD_INTRASTATE	320	/*Network Access Fee LD-Intrastate*/
#define VENDOR_USE_TAX	321	/*Vendor Use Tax*/
#define DISTRICT_VENDOR_USE_TAX	322	/*District Vendor Use Tax*/
#define SPECIAL_VENDOR_USE_TAX	323	/*Special Vendor Use Tax*/
#define TRANSIT_VENDOR_USE_TAX	324	/*Transit Vendor Use Tax*/
#define CRIME_CONTROL_DISTRICT_VENDOR_USE_TAX	325	/*Crime Control District Vendor Use Tax*/
#define LIBRARY_DISTRICT_VENDOR_USE_TAX	326	/*Library District Vendor Use Tax*/
#define HOSPITAL_DISTRICT_VENDOR_USE_TAX	327	/*Hospital District Vendor Use Tax*/
#define HEALTH_SERVICES_DISTRICT_VENDOR_USE_TAX	328	/*Health Services District Vendor Use Tax*/
#define EMERGENCY_SERVICES_DISTRICT_VENDOR_USE_TAX	329	/*Emergency Services District Vendor Use Tax*/

#define IMPROVEMENT_DISTRICT_VENDOR_USE_TAX	330	/*Improvement
District Vendor Use Tax*/		
#define DEVELOPMENT_DISTRICT_VENDOR_USE_TAX	331	/*Development
District Vendor Use Tax*/		
#define AMBULANCE_DISTRICT_VENDOR_USE_TAX	332	/*Ambulance
District Vendor Use Tax*/		
#define FIRE_DISTRICT_VENDOR_USE_TAX	333	/*Fire District
Vendor Use Tax*/		
#define FOOTBALL_DISTRICT_VENDOR_USE_TAX	334	/*Football
District Vendor Use Tax*/		
#define BASEBALL_DISTRICT_VENDOR_USE_TAX	335	/*Baseball
District Vendor Use Tax*/		
#define EDUCATIONAL_VENDOR_USE_TAX	336	/*Educational
Vendor Use Tax*/		
#define SCHOOL_DISTRICT_VENDOR_USE_TAX	337	/*School District
Vendor Use Tax*/		
#define ADVANCED_TRANSIT_VENDOR_USE_TAX	338	/*Advanced
Transit Vendor Use Tax*/		
#define TRIBAL_VENDOR_USE_TAX	339	/*Tribal Vendor
Use Tax*/		
#define VENDOR_USE_TAX_SENIOR_CITIZEN	340	/*Vendor Use Tax-
Senior Citizen*/		
#define VENDOR_USE_TAX_MANUFACTURING	341	/*Vendor Use Tax-
Manufacturing*/		
#define VENDOR_USE_TAX_MOTOR_VEHICLES	342	/*Vendor Use Tax-
Motor Vehicles*/		
#define VENDOR_USE_TAX_VENDING	343	/*Vendor Use Tax-
Vending*/		
#define VENDOR_USE_TAX_FOOD_AND_DRUGS	344	/*Vendor Use Tax-
Food and Drugs*/		
#define VENDOR_USE_TAX_FOOD	345	/*Vendor Use Tax-
Food*/		
#define VENDOR_USE_TAX_MOTOR_FUEL	346	/*Vendor Use Tax-
Motor Fuel*/		
#define VENDOR_USE_TAX_PARKING	347	/*Vendor Use Tax-
Parking*/		
#define VENDOR_USE_TAX_MEDICAL_EQUIPMENT	348	/*Vendor Use Tax-
Medical Equipment*/		
#define ALCOHOLIC_BEVERAGE_VENDOR_USE_TAX	349	/*Alcoholic
Beverage Vendor Use Tax*/		
#define VENDOR_USE_TAX_ALCOHOL	350	/*Vendor Use Tax-
Alcohol*/		
#define LIQUOR_DRINK_VENDOR_USE_TAX	351	/*Liquor Drink
Vendor Use Tax*/		
#define VENDOR_USE_TAX_VIDEO	352	/*Vendor Use Tax-
Video*/		
#define PREMIER_RESORT_AREA_VENDOR_USE_TAX	353	/*Premier Resort
Area Vendor Use Tax*/		
#define NY_TRANSIT_VENDOR_USE_TAX	354	/*NY Transit
Vendor Use Tax*/		
#define NY_DISTRICT_VENDOR_USE_TAX	355	/*NY District
Vendor Use Tax*/		
#define VENDOR_USE_TAX_FOOD_AND_BEVERAGE	356	/*Vendor Use Tax-
Food and Beverage*/		
#define CONSUMER_USE_TAX	357	/*Consumer Use
Tax*/		
#define DISTRICT_CONSUMER_USE_TAX	358	/*District
Consumer Use Tax*/		
#define SPECIAL_CONSUMER_USE_TAX	359	/*Special
Consumer Use Tax*/		
#define TRANSIT_CONSUMER_USE_TAX	360	/*Transit
Consumer Use Tax*/		
#define CRIME_CONTROL_DISTRICT_CONSUMER_USE_TAX	361	/*Crime Control
District Consumer Use Tax*/		
#define LIBRARY_DISTRICT_CONSUMER_USE_TAX	362	/*Library
District Consumer Use Tax*/		
#define HOSPITAL_DISTRICT_CONSUMER_USE_TAX	363	/*Hospital
District Consumer Use Tax*/		
#define HEALTH_SERVICES_DISTRICT_CONSUMER_USE_TAX	364	/*Health Services
District Consumer Use Tax*/		

#define EMERGENCY_SERVICES_DISTRICT_CONSUMER_USE_TAX	365	/*Emergency
Services District Consumer UseTax*/		
#define IMPROVEMENT_DISTRICT_CONSUMER_USE_TAX	366	/*Improvement
District Consumer Use Tax*/		
#define DEVELOPMENT_DISTRICT_CONSUMER_USE_TAX	367	/*Development
District Consumer Use Tax*/		
#define AMBULANCE_DISTRICT_CONSUMER_USE_TAX	368	/*Ambulance
District Consumer Use Tax*/		
#define FIRE_DISTRICT_CONSUMER_USE_TAX	369	/*Fire District
Consumer Use Tax*/		
#define FOOTBALL_DISTRICT_CONSUMER_USE_TAX	370	/*Football
District Consumer Use Tax*/		
#define BASEBALL_DISTRICT_CONSUMER_USE_TAX	371	/*Baseball
District Consumer Use Tax*/		
#define EDUCATIONAL_CONSUMER_USE_TAX	372	/*Educational
Consumer Use Tax*/		
#define SCHOOL_DISTRICT_CONSUMER_USE_TAX	373	/*School District
Consumer Use Tax*/		
#define ADVANCED_TRANSIT_CONSUMER_USE_TAX	374	/*Advanced
Transit Consumer Use Tax*/		
#define TRIBAL_CONSUMER_USE_TAX	375	/*Tribal Consumer
Use Tax*/		
#define CONSUMER_USE_TAX_SENIOR_CITIZEN	376	/*Consumer Use
Tax-Senior Citizen*/		
#define CONSUMER_USE_TAX_MANUFACTURING	377	/*Consumer Use
Tax-Manufacturing*/		
#define CONSUMER_USE_TAX_MOTOR_VEHICLES	378	/*Consumer Use
Tax-Motor Vehicles*/		
#define CONSUMER_USE_TAX_VENDING	379	/*Consumer Use
Tax-Vending*/		
#define CONSUMER_USE_TAX_FOOD_AND_DRUGS	380	/*Consumer Use
Tax-Food and Drugs*/		
#define CONSUMER_USE_TAX_FOOD	381	/*Consumer Use
Tax-Food*/		
#define CONSUMER_USE_TAX_MOTOR_FUEL	382	/*Consumer Use
Tax-Motor Fuel*/		
#define CONSUMER_USE_TAX_PARKING	383	/*Consumer Use
Tax-Parking*/		
#define CONSUMER_USE_TAX_MEDICAL_EQUIPMENT	384	/*Consumer Use
Tax-Medical Equipment*/		
#define ALCOHOLIC_BEVERAGE_CONSUMER_USE_TAX	385	/*Alcoholic
Beverage Consumer Use Tax*/		
#define CONSUMER_USE_TAX_ALCOHOL	386	/*Consumer Use
Tax-Alcohol*/		
#define LIQUOR_DRINK_CONSUMER_USE_TAX	387	/*Liquor Drink
Consumer Use Tax*/		
#define CONSUMER_USE_TAX_VIDEO	388	/*Consumer Use
Tax-Video*/		
#define PREMIER_RESORT_AREA_CONSUMER_USE_TAX	389	/*Premier Resort
Area Consumer Use Tax*/		
#define NY_LOCAL_TRANSIT_CONSUMER_USE_TAX	390	/*NY Transit
Consumer Use Tax*/		
#define NY_DISTRICT_CONSUMER_USE_TAX	391	/*NY District
Consumer Use Tax*/		
#define CONSUMER_USE_TAX_FOOD_AND_BEVERAGE	392	/*Consumer Use
Tax-Food and Beverage*/		
#define TASA_DE_CONTROL	393	/*Tasa de
Control*/		
#define RADIO_RIGHTS_FEE	394	/*Radio Rights
Fee*/		
#define BUSINESS_AND_OCCUPATION_TAX_RENT_AND_ROYALTY	395	/*Business and
Occupation Tax-Rent and Royalty*/		
#define BUSINESS_AND_OCCUPATION_TAX_OTHER_SERVICES	396	/*Business and
Occupation Tax-Other Services*/		
#define MONTANA_EXCISE_TAX	397	/*Montana Excise
Tax*/		
#define RURAL_TRANSPORTATION_AUTHORITY_DISTRICT_TAX	398	/*Rural
Transportation Authority District Tax*/		
#define MHA_DISTRICT_TAX	399	/*MHA District
Tax*/		

#define PUBLIC_SAFETY_IMPROVEMENTS_DISTRICT_TAX	400	/*Public Safety
Improvements District Tax*/		
#define MASS_TRANSIT_DISTRICT_TAX	401	/*Mass Transit
District Tax*/		
#define METROPOLITAN_DISTRICT_TAX	402	/*Metropolitan
District Tax*/		
#define RTA_CONSUMER_USE_TAX	403	/*RTA Consumer
Use Tax*/		
#define RTA_VENDOR_USE_TAX	404	/*RTA Vendor Use
Tax*/		
#define MHA_CONSUMER_USE_TAX	405	/*MHA Consumer
Use Tax*/		
#define MHA_VENDOR_USE_TAX	406	/*MHA Vendor Use
Tax*/		
#define MASS_TRANSIT_DISTRICT_CONSUMER_USE_TAX	407	/*Mass Transit
District Consumer Use Tax*/		
#define MASS_TRANSIT_DISTRICT_VENDOR_USE_TAX	408	/*Mass Transit
District Vendor Use Tax*/		
#define VAT_REDUCED_RATE	409	/*VAT (Reduced
Rate)*/		
#define POISON_CONTROL_FUND_WIRELESS	410	/*Poison Control
Fund (Wireless)*/		
#define STATE_INSPECTION_AND_SUPERVISION	411	/*State
Inspection and Supervision*/		
#define EDUCATION_SALES_VENDING	412	/*Education
Sales-Vending*/		
#define EDUCATION_SALES_MOTOR_VEHICLES	413	/*Education
Sales-Motor Vehicles*/		
#define EDUCATION_USE_MOTOR_VEHICLES	414	/*Education Use-
Motor Vehicles*/		
#define EDUCATION_CONSUMER_USE_MOTOR_VEHICLES	415	/*Education
Consumer Use-Motor Vehicles*/		
#define EDUCATION_VENDOR_USE_MOTOR_VEHICLES	416	/*Education
Vendor Use-Motor Vehicles*/		
#define EDUCATION_SALES_MANUFACTURING	417	/*Education
Sales-Manufacturing*/		
#define EDUCATION_USE_MANUFACTURING	418	/*Education Use-
Manufacturing*/		
#define EDUCATION_CONSUMER_USE_MANUFACTURING	419	/*Education
Consumer Use - Manufacturing*/		
#define EDUCATION_VENDOR_USE_MANUFACTURING	420	/*Education
Vendor Use - Manufacturing*/		
#define RENTAL_USE_TAX_MOTOR_VEHICLES	421	/*Rental Use Tax
- Motor Vehicles*/		
#define CONSUMER_USE_RENTAL_TAX_MOTOR_VEHICLES	422	/*Consumer Use
Rental Tax - Motor Vehicles*/		
#define VENDOR_USE_RENTAL_TAX_MOTOR_VEHICLES	423	/*Vendor Use
Rental Tax - Motor Vehicles*/		
#define REVENUE_STATEMENT	424	/*Revenue
Statement*/		
#define NY_MCTD_186C_WIRELESS	425	/*NY MCTD 186c
(Wireless)*/		
#define NY_MCTD_186C_WIRELESS	425	/*NY MCTD 186c
(Wireless)*/		
#define WY_USF	426	/*WY USF*/
#define WY_USF_PAGING	427	/*WY USF
(Paging)*/		
#define WY_USF_WIRELESS	428	/*WY USF
(Wireless)*/		
#define FCC_REGULATORY_FEE_TOLL_FREE	429	/*FCC Regulatory
Fee-Toll Free*/		
#define FCC_REGULATORY_FEE_SATELLITE	430	/*FCC Regulatory
Fee (Satellite)*/		
#define COMMERCE_TAX	431	/*Commerce Tax*/
#define TELECOM_ASSISTANCE_SVC_FUND_VOIP	432	/*Telecom
Assistance Svc Fund - VoIP*/		
#define TELECOM_ASSISTANCE_SVC_FUND_VOIP_HIGH_CAP_TRNK	433	/*Telecom
Assistance Svc Fund - VoIP High Cap Trnk*/		
#define E911_VOIP_NOMADIC_PBX	434	/*E-911 (VoIP-
Nomadic PBX)*/		

#define E911_SERVICE_FEE_NL_911_BUREAU	435	/*E-911 Service
Fee (NL 911 Bureau)*/		
#define COPYRIGHT_FEE_RATED	436	/*Copyright Fee
(Rated) */		
#define COPYRIGHT_FEE_FIXED	437	/*Copyright Fee
(Fixed) */		
#define UTILITY_TAX	438	/*Utility Tax*/
#define AUDIO_VIDEO_SERVICE_TAX	439	/*Audio-Video
Service Tax*/		
#define SWACHH_BHARAT_CESS	440	/*Swachh Bharat
Cess*/		
#define PIS	441	/*PIS*/
#define COFINS	442	/*COFINS*/
#define ICMS	443	/*ICMS*/
#define FEDERAL_USF_CENTREX	444	/*Federal USF
(Centrex) */		
#define UUT_PREPAID_WIRELESS	445	/*UUT (Prepaid
Wireless) */		
#define MOBILE_TELEPHONY_SERVICES_SURCHARGE	446	/*Mobile
Telephony Services Surcharge*/		
#define ACCESS_LINE_TAX_PREPAID_WIRELESS	447	/*Access Line Tax
(Prepaid Wireless) */		
#define SAN_LEANDRO_EMERG_COM_SYS_ACC_TAX_PPD_WIRELESS	448	/*San Leandro
Emerg Com Sys Acc Tax(Ppd Wireless) */		
#define RENTAL_TAX_LOWER_RATE	449	/*Rental Tax
(Lower Rate) */		
#define CA_HIGH_COST_FUND_A_VOIP	450	/*CA High Cost
Fund A (VoIP) */		
#define STATE_HIGH_COST_FUND_VOIP	451	/*State High Cost
Fund (VoIP) */		
#define CA_TELECONNECT_FUND_VOIP	452	/*CA Teleconnect
Fund (VoIP) */		
#define CASF_VOIP	453	/*CASF (VoIP) */
#define UNIVERSAL_LIFELINE_TELEPHONE_SERVICE_CHARGE_VOIP	454	/*Universal
Lifeline Telephone Service Charge (VoIP) */		
#define FUNTTEL	455	/*FUNTTEL*/
#define FUST	456	/*FUST*/
#define TELECOMMUNICATIONS_USE_TAX	457	/*Telecommunications Use Tax*/
ons Use Tax*/		
#define KRISHI_KALYAN_CESS	458	/*Krishi Kalyan
Cess*/		
#define SCHOOL_AND_LIBRARY_FUND_SURCHARGE	459	/*School and
Library Fund Surcharge*/		
#define STATE_911_CHARGE	460	/*State 911
Charge*/		
#define ITAC_ASSESSMENT	461	/*ITAC
Assessment*/		
#define STATE_911_CHARGE_WIRELESS	462	/*State 911
Charge (Wireless) */		
#define E911_ADVANCED_SERVICES	463	/*E-911 (Advanced
Services) */		
#define VAT_WIRELESS	464	/*VAT
(Wireless) */		
#define VAT_COMMUNICATIONS	465	/*VAT
(Communications) */		
#define CA_TRS	466	/*CA TRS*/
#define CA_TRS_WIRELESS	467	/*CA TRS
(Wireless) */		
#define CA_PUC_FEE	468	/*CA PUC Fee*/
#define USE_TAX_RENTAL	469	/*Use Tax
(Rental) */		
#define USE_TAX_OTHER	470	/*Use Tax
(Other) */		
#define CONSUMER_USE_TAX_OTHER	471	/*Consumer Use
Tax (Other) */		
#define VENDOR_USE_TAX_OTHER	472	/*Vendor Use Tax
(Other) */		
#define SC_USF	473	/*SC USF*/
#define USF_PREPAID_WIRELESS	474	/*USF (Prepaid
Wireless) */		

#define E911_LIFELINE	475	/*E-911
(Lifeline)*/		
#define UTILITY_TAX_NF	476	/*Utility Tax
NF*/		
#define TELECOMMUNICATIONS_SALES_TAX_WHOLESALES	477	/*Telecommuni
cations Sales Tax (Wholesale)*/		
#define E_RATE_BROADBAND_PROGRAM	478	/*E-rate
Broadband Program*/		
#define E_RATE_BROADBAND_PROGRAM_BUSINESS_LINE	479	/*E-rate
Broadband Program (Business Line)*/		
#define E_RATE_BROADBAND_PROGRAM_LINE	480	/*E-rate
Broadband Program (Line)*/		
#define E_RATE_BROADBAND_PROGRAM_WIRELESS	481	/*E-rate
Broadband Program (Wireless)*/		
#define IGST_COMMUNICATIONS	482	/*IGST
(Communications)*/		
#define CGST	483	/*CGST*/
#define CGST_COMMUNICATIONS	484	/*CGST
(Communications)*/		
#define SGST	485	/*SGST*/
#define SGST_COMMUNICATIONS	486	/*SGST
(Communications)*/		
#define UNIVERSAL_SERVICE_FUND_OTHER	487	/*Universal
Service Fund (Other)*/		
#define IGST	488	/*IGST*/
#define KENTUCKY_LIFELINE_SURCHARGE	489	/*Kentucky
Lifeline Surcharge*/		
#define TELECOMMUNICATIONS_SALES_TAX_NF	490	/*Telecommuni
cations Sales Tax NF*/		
#define PUBLIC_SAFETY_COMMUNICATIONS_SURCHARGE_PREPAID	491	/*Public Safety
Communications' Surcharge (Prepaid)*/		
#define STATUTORY_GROSS_RECEIPTS_NF	492	/*Statutory Gross
Receipts NF*/		

#define TELECOMM_SALE_TAX	32711	/*For BillSoft
internal use only*/		

/* Deprecated Tax Type constants */		
#define BUSINESS_OCCUPATION_TAX		BUSINESS_AND_OCCUPATION_TAX
#define FEDERAL_EXCISE		FEDERAL_EXCISE_TAX
#define FED_USF SCHOOL_A		FED_USF_A_SCHOOL
#define E911_TAX		E911
#define STATE_USF		STATE_UNIVERSAL_SERVICE_FUND
#define FED_USF_HIGHCOST_B		FED_UNIVERSAL_SERVICE_FUND
#define STATE_DEAF_DISABLED_FUND		STATE_DEAF_AND_DISABLED_FUND
#define TELECOMMUNICATIONS_RELAY_SERVICE_CHARGE		TELECOM_RELAY_SURCHARGE
#define TELECOM_RELAY_CHARGE		TELECOM_RELAY_SURCHARGE
#define TELECOMMUNICATIONS_INFRASTRUCTURE_FUND		
TELECOMMUNICATIONS_INFRASTRUCTURE_FUND		
#define NY_MCTD_186c		NY_MCTD_186C
#define NY_MCTD_184a		NY_MCTD_184A
#define TELECOMMUNICATIONS_ASSISTANCE_FUND		
TELECOMMUNICATIONS_ASSISTANCE_SERVICE_FUND		
#define E911_TAX_PBX_TRUNK_LINE		E911_PBX_TRUNK_LINE
#define E911_TAX_RESIDENTIAL		E911_RESIDENTIAL
#define E911_TAX_WIRELESS		E911_WIRELESS
#define NY_MCTD_184a_USAGE		NY_MCTD_184A_USAGE
#define MUNICIPAL_RIGHT_OF WAY_RESIDENTIAL		MUNICIPAL_RIGHT_OF WAY
#define COMMUNICATIONS_SERVICES_TAX		COMMUNICATIONS_SERVICE_TAX
#define GOODS_SERVICE_TAX		GOODS_AND_SERVICE_TAX
#define SGT_E911		SGT_E911_TAX
#define SGT_E911_BUSINESS		SGT_E911_TAX_BUSINESS
#define SGT_E911_LICENSE_TAX		SGT_LICENSE_TAX
#define FRANCISE_TAX_WIRELESS		FRANCHISE_TAX_WIRELESS
#define E911_TAX_CENTREX		E911_CENTREX
#define BUSINESS_OCCUPATION_TAX_WHOLESALE		BUSINESS_AND_OCCUPATION_TAX_WHOLESALE

```

#define BUSINESS_OCCUPATION_TAX_OTHER
#define STATE_USF_VOIP
#define COMMUNICATIONS_SERVICES_TAX_CABLE
#define TRS_WIRELESS
#define TELECOM_RELAY_CHARGE_WIRELESS
#define CONNECT_ME_FUND
#define CONNECT_ME_FUND_VOIP
#define CONNECT_ME_FUND_CABLE
#define FUSF_NONBILLABLE
FEDERAL_UNIVERSAL_SERVICE_FUND_NON_BILLABLE
#define UUT_WIRELESS_BUSINESS
#define STATE_UNIV_SVC_FUND_ALTERNATE
#define UNIV_LIFELINE_TELE_SVC_CHARGE_WIRELESS
UNIVERSAL_LIFELINE_TELEPHONE_SVC_CHARGE_WIRELESS
#define FUSF_MULTI_LINE
#define NY_TRANSIT_CONSUMER_USE_TAX

#ifndef __cplusplus
#endif
#endif /* _inc_EZTaxTaxTypeDefine_ */

```

BUSINESS_AND_OCCUPATION_TAX_OTHER
UNIVERSAL_SERVICE_FUND_VOIP
COMMUNICATIONS_SERVICE_TAX_CABLE
TELECOM_RELAY_SURCHARGE_WIRELESS
TELECOM_RELAY_SURCHARGE_WIRELESS
CONNECTME_FUND
CONNECTME_FUND_VOIP
CONNECTME_FUND_CABLE

UTILITY_USERS_TAX_WIRELESSBUSINESS
GA_UNIVERSAL_ACCESS_FUND

FUSF_MULTI_LINE
NY_LOCAL_TRANSIT_CONSUMER_USE_TAX

```
EZTaxTransType.h
```

```
#ifndef _inc_EZTaxTransTypeDefine_
#define _inc_EZTaxTransTypeDefine_

#ifndef __cplusplus
extern "C" {
#endif

/* Define transaction types */
#define TRANS_NO_TAX 0 /*No Tax*/
#define INTERSTATE 1 /*Interstate*/
#define INTRASTATE 2 /*Intrastate*/
#define OTHER 3 /*Other*/
#define NON_RECURRING 4 /*Non-Recurring*/
#define INTERNET 5 /*Internet*/
#define PAGING 6 /*Paging*/
#define LOCAL 7 /*Local*/
#define FAX 8 /*Fax*/
#define VOICE_MAIL 9 /*Voice Mail*/
#define SALES 10 /*Sales*/
#define SHIPPING 11 /*Shipping*/
#define NATURAL_GAS 12 /*Natural Gas*/
#define CELLULAR 13 /*Cellular*/
#define INTERNATIONAL 14 /*International*/
#define TELEPHONY 15 /*Telephony*/
#define CABLE_TELEVISION 16 /*Cable Television*/
#define SGT 17 /*Sage*/
#define SATELLITE_TELEVISION 18 /*Satellite Television*/
#define VOIP 19 /*VoIP*/
#define VOIPA 20 /*VoIPA*/
#define PAYPHONE 21 /*Payphone*/
#define SOFTWARE 24 /*Software*/
#define TIMESHARING 25 /*Timesharing*/
#define ALCOHOL 27 /*Alcohol*/
#define BEVERAGES 28 /*Beverages*/
#define BOOKS 29 /*Books*/
#define CLOTHING 30 /*Clothing*/
#define DRUGS 31 /*Drugs*/
#define ELECTRONIC_EQUIPMENT_AND_COMPUTER_HARDWARE 32 /*Electronic Equipment and Computer Hardware*/
#define FUEL 33 /*Fuel*/
#define GENERAL_MERCHANDISE 34 /*General Merchandise*/
#define GROCERIES 35 /*Groceries*/
#define MAGAZINES 36 /*Magazines*/
#define MANUFACTURING 37 /*Manufacturing*/
#define MEDICAL_DURABLE_EQUIPMENT 38 /*Medical-Durable Equipment*/
#define MEDICAL_MOBILITY_ENHANCING_EQUIPMENT 39 /*Medical-Mobility Enhancing Equipment*/
#define MEDICAL_PROSTHETIC_DEVICES 40 /*Medical-Prosthetic Devices*/
#define MOTOR_VEHICLES 41 /*Motor Vehicles*/
#define NEWSPAPER 42 /*Newspaper*/
#define PREPARED_FOOD 43 /*Prepared Food*/
#define RENTALS_AND_LEASING 44 /*Rentals and Leasing*/
#define SERVICES_CLEANING 45 /*Services-Cleaning*/
#define SERVICES LODGING 46 /*Services-Lodging*/
#define SERVICES_PRINTING 47 /*Services-Printing*/
#define SERVICES_PROFESSIONAL 48 /*Services-Professional*/

#endif
```

```

#define SERVICES_CREATION 49 /*Services-
Recreation*/
#define SERVICES_REPAIR 50 /*Services-
Repair*/
#define SERVICES_STORAGE 51 /*Services-
Storage*/
#define TIRES 52 /*Tires*/
#define TOBACCO 53 /*Tobacco*/
#define TOOLING 54 /*Tooling*/
#define VENDING 55 /*Vending*/
#define INFORMATION_SERVICES 56 /*Information
Services*/
#define DIGITAL_GOODS 57 /*Digital Goods*/
#define DARK_FIBER 58 /*Dark Fiber*/
#define VOIP_NOMADIC 59 /*VoIP-Nomadic*/
#define SATELLITE_PHONE 60 /*Satellite
Phone*/
#define VPN 61 /*VPN*/
#define RESERVED_62 62 /*Reserved_62*/
#define RESERVED_63 63 /*Reserved_63*/
#define CONFERENCING 64 /*Conferencing*/
#define NON_INTERCONNECTED_VOIP 65 /*Non-
Interconnected VoIP*/

/* Deprecated transaction type constants */
#define DO_NOT_APPLY_TAX NO_TAX
#define INTERPROVINCIAL INTERSTATE
#define INTRAPROVINCIAL INTRASTATE
#define WIRELESS CELLULAR

#ifndef __cplusplus
#endif

#endif /* _inc_EZTaxTransTypeDefine_ */

```

```

EZTaxServType.h

#ifndef _inc_EZTaxServTypeDefine_
#define _inc_EZTaxServTypeDefine_

#ifndef __cplusplus
extern "C" {
#endif

/* Define service types */
#define TOLL 1 /*Toll*/
#define TOLL_FREE 2 /*Toll-Free*/
#define WATS 3 /*WATS*/
#define PRIVATE_LINE 4 /*Private Line*/
#define LOCAL_EXCHANGE 5 /*Local Exchange*/
#define ACCESS_CHARGE 6 /*Access Charge*/
#define SERVICE 7 /*Service*/
#define INSTALL 8 /*Install*/
#define DIRECTORY_ADS 9 /*Directory Ads*/
#define USAGE 10 /*Usage*/
#define ACTIVATION 11 /*Activation*/
#define INTERNATIONAL_TOLL 12 /*International Toll*/
#define EQUIPMENT_REPAIR 13 /*Equipment Repair*/
#define LATE_CHARGE 14 /*Late Charge*/
#define PRODUCT 15 /*Product*/
#define N_900_SERVICE 16 /*900*/
#define FOB_ORIGIN 17 /*FOB Origin*/
#define FOB_DESTINATION 18 /*FOB Destination*/
#define CONSUMPTION 19 /*Consumption*/
#define FCC_SUBSCRIBER_LINE_FEE 20 /*FCC Subscriber Line Fee*/
#define LINES 21 /*Lines*/
#define COIN 22 /*Coin*/
#define LOCATION 23 /*Location*/
#define PBX_TRUNK 24 /*PBX/Trunk*/
#define USA_INBOUND 25 /*USA Inbound*/
#define PREPAID 26 /*Prepaid*/
#define DATA 27 /*Data*/
#define E911_CALL 28 /*E911 Call*/
#define WEB_HOSTING 29 /*WEB Hosting*/
#define LOCAL_FEATURE_CHARGE 30 /*Local Feature Charge*/
#define USE 31 /*Use*/
#define DEBIT 32 /*Debit*/
#define ROAMING_CHARGE 33 /*Roaming Charge*/
#define CONFERENCE_BRIDGE 34 /*Conference Bridge*/
#define PREMIUM_SERVICE 35 /*Premium Service*/
#define PAY_PER_VIEW_SERVICE 36 /*Pay Per View Service*/
#define EQUIPMENT_RENTAL 37 /*Equipment Rental*/
#define WIRE_MAINTENANCE_PLAN 38 /*Wire Maintenance Plan*/
#define TV_GUIDE 39 /*TV Guide*/
#define CENTREX_DID_EXTENSION 40 /*Centrex/DID Extension*/

```

```

#define PBX_EXTENSION 41 /*PBX
Extension*/
#define CENTREX_TRUNK 42 /*Centrex
Trunk*/
#define INVOICE 43 /*Invoice*/
#define TELEGRAPH 44 /*Telegraph*/
#define HIGH_CAPACITY_TRUNK 45 /*High
Capacity Trunk*/
#define PICC 46 /*PICC*/
#define NO_PICK_PICC 47 /*No Pick
PICC*/
#define WIRELESS_ACCESS_CHARGE 48 /*Wireless
Access Charge*/
#define INTERSTATE_USAGE 49 /*Interstate
Usage*/
#define INTRASTATE_USAGE 50 /*Intrastate
Usage*/
#undef INTERNATIONAL_USAGE 51 /*International Usage*/
#define INTERNATIONAL_USAGE 52 /*Wireless
Lines*/
#define LNP 53 /*LNP*/
#define DIRECTORY_ASSISTANCE 54 /*Directory
Assistance*/
#define LOCAL_USAGE 55 /*Local
Usage*/
#define PROVISIONING 56 /*Provisioning*/
#define DATA_PROCESSING 57 /*Data
Processing*/
#define ACCESS_LINE 58 /*Access
Line*/
#define LICENSED_SOFTWARE 59 /*Licensed
Software*/
#define SOFTWARE_MAINTENANCE AGREEMENT 60 /*Software
Maintenance Agreement*/
#define REPORT_ON_CD_PAPER_FORM 61 /*Report On
CD/Paper Form*/
#define INFORMATION_RETRIEVAL 62 /*Information
Retrieval*/
#define RESTOCKING_FEE_RENTAL 63 /*Restocking
Fee - Rental*/
#define RESTOCKING_FEE_PURCHASE 64 /*Restocking
Fee - Purchase*/
#define PARTIAL_CREDIT 65 /*Partial
Credit*/
#define LATE_CHARGE_BUNDLE 84 /*Late Charge
Bundle*/
#define LOCAL_EXCHANGE_BUNDLE 85 /*Local
Exchange Bundle*/
#define FCC_SUBSCRIBER_LINE_FEE_BUNDLE 86 /*FCC
Subscriber Line Fee Bundle*/
#define LINES_BUNDLE 87 /*Lines
Bundle*/
#define LOCATION_BUNDLE 88 /*Location
Bundle*/
#define PBX_TRUNK_BUNDLE 89 /*PBX Trunk
Bundle*/
#define LOCAL_FEATURE_CHARGE_BUNDLE 90 /*Local
Feature Charge Bundle*/
#define CENTREX_EXTENSION_BUNDLE 91 /*Centrex
Extension Bundle*/
#define PBX_EXTENSION_BUNDLE 92 /*PBX
Extension Bundle*/
#define CENTREX_TRUNK_BUNDLE 93 /*Centrex
Trunk Bundle*/
#define INVOICE_BUNDLE 94 /*Invoice
Bundle*/
#define HIGH_CAPACITY_TRUNK_BUNDLE 95 /*High
Capacity Trunk Bundle*/

```

#define NO_PICK_PICC_BUNDLE	96	/*No Pick PICC
Bundle*/		
#define PICC_BUNDLE	97	/*PICC
Bundle*/		
#define ACCESS_NUMBER	98	/*Access
Number*/		
#define INTERSTATE_ACCESS_CHARGE	99	/*Interstate
Access Charge*/		
#define INTRASTATE_ACCESS_CHARGE	100	/*Intrastate
Access Charge*/		
#define INTERSTATE_ROAMING	101	/*Interstate
Roaming*/		
#define INTRASTATE_ROAMING	102	/*Intrastate
Roaming*/		
#define SALES_TAX_AND_FUSF	103	/*Sales Tax
and FUSF*/		
#define GENERAL_RULE	106	/*General
Rule*/		
#define BEVERAGE_ABOVE_7_PCT_CONTENT_BY_WEIGHT	107	/*Beverage
Above 7% Content By Weight*/		
#define BEVERAGE_BELOW_7_PCT_CONTENT_BY_WEIGHT	108	/*Beverage
Below 7% Content By Weight*/		
#define MIXED_BEVERAGE_ABOVE_7_PCT_CONTENT_BY_WEIGHT	109	/*Mixed
Beverage above 7% Content By Weight*/		
#define MIXED_BEVERAGE_BELOW_7_PCT_CONTENT_BY_WEIGHT	110	/*Mixed
Beverage below 7% Content By Weight*/		
#define NON_MIXED_SERVED_IN_RESTAURANT_ABOVE_7_PCT	111	/*Non Mixed
Served in Restaurant Above 7%*/		
#define NON_MIXED_SERVED_IN_RESTAURANT_BELOW_7_PCT	112	/*Non Mixed
Served in Restaurant Below 7%*/		
#define CARBONATED_BEVERAGES	113	/*Carbonated
Beverages*/		
#define SWEETENED_CARBONATED_BEVERAGES	114	/*Sweetened
Carbonated Beverages*/		
#define BOTTLED_WATER	115	/*Bottled
Water*/		
#define BOTTLED_WATER_CARBONATED_AND_OR_FLAVORED	116	/*Bottled
Water - Carbonated and/or Flavored*/		
#define BOTTLED_WATER_CARBONATED_AND_OR_SWEETENED	117	/*Bottled
Water-Carbonated and/or Sweetened*/		
#define SOFT_DRINKS	118	/*Soft
Drinks*/		
#define NATURAL_FRUIT_OR_VEGETABLE_JUICES	119	/*Natural
Fruit or Vegetable Juices*/		
#define NATURAL_CONTENTS_BETWEEN_0_PCT_24_PCT	120	/*Natural
Contents Between 0%-24%*/		
#define NATURAL_CONTENTS_BETWEEN_25_PCT_49_PCT	121	/*Natural
Contents Between 25%-49%*/		
#define NATURAL_CONTENTS_BETWEEN_50_PCT_69_PCT	122	/*Natural
Contents Between 50%-69%*/		
#define NATURAL_CONTENTS_BETWEEN_70_PCT_100_PCT	123	/*Natural
Contents Between 70%-100%*/		
#define BOTTLED_TEA	124	/*Bottled
Tea*/		
#define COFFEE	125	/*Coffee*/
#define RELIGIOUS	126	/*Religious*/
#define EDUCATIONAL_KINDERGARTEN_THROUGH_12TH_GRADE	127	/*Educational-
Kindergarten Through 12th Grade*/		
#define EDUCATIONAL_COLLEGE_AND_TRADE SCHOOL	128	/*Educational-
College and Trade School*/		
#define EVERYDAY	129	/*Everyday*/
#define SPORTING_ACTIVITIES	130	/*Sporting
Activities*/		
#define SPORTING_EQUIPMENT	131	/*Sporting
Equipment*/		
#define PROTECTIVE	132	/*Protective*/
#define PROTECTIVE_MANUFACTURING	133	
/*Protective/Manufacturing*/		
#define FURS	134	/*Furs*/
#define UNIFORMS	135	/*Uniforms*/

#define FORMAL_OR_SPECIAL_OCCASION_WEAR	136	/*Formal or Special Occasion Wear*/
#define COSTUMES	137	/*Costumes*/
#define ACCESSORIES	138	/*Accessories*/
#define DISPLAY_SAMPLES	139	/*Display Samples*/
#define CLOTH_DIAPERS	140	/*Cloth Diapers*/
#define CLEAN_ROOM	141	/*Clean Room*/
#define BATHING_CAPS	142	/*Bathing Caps*/
#define BELT_BUCKLES	143	/*Belt Buckles*/
#define BOWLING_SHOES	144	/*Bowling Shoes*/
#define SKI_BOOTS	145	/*Ski Boots*/
#define WADERS	146	/*Waders*/
#define SHOE_LACES	147	/*Shoe Laces*/
#define PRESCRIPTION_LEGEND	148	/*Prescription Legend*/
#define PRESCRIPTION_OVER_THE_COUNTER_HUMAN	149	/*Prescription - Over the Counter-Human*/
#define NONPRESCRIPTION_OVER_THE_COUNTER_HUMAN	150	/*Nonprescription - Over the Counter-Human*/
#define PRESCRIPTION_OVER_THE_COUNTER_ANIMAL	151	/*Prescription - Over the Counter-Animal*/
#define NONPRESCRIPTION_OVER_THE_COUNTER_ANIMAL	152	/*Nonprescription - Over the Counter-Animal*/
#define COUGH_DROPS	153	/*Cough Drops*/
#define PRESCRIPTION_INSULIN_HUMAN_USE	154	/*Prescription - Insulin - Human Use*/
#define NONPRESCRIPTION_INSULIN_HUMAN_USE	155	/*Nonprescription - Insulin - Human Use*/
#define PRESCRIPTION_INSULIN_ANIMAL_USE	156	/*Prescription - Insulin - Animal Use*/
#define NONPRESCRIPTION_INSULIN_ANIMAL_USE	157	/*Nonprescription - Insulin - Animal Use*/
#define PRESCRIPTION_OXYGEN_HUMAN_USE	158	/*Prescription - Oxygen-Human Use*/
#define NONPRESCRIPTION_OXYGEN_MEDICINAL_HUMAN_USE	159	/*Nonprescription - Oxygen-Medicinal-Human Use*/
#define PRESCRIPTION_OXYGEN_ANIMAL_USE	160	/*Prescription - Oxygen-Animal Use*/
#define NONPRESCRIPTION_OXYGEN_MEDICINAL_ANIMAL_USE	161	/*Nonprescription-Oxygen-Medicinal-Animal Use*/
#define ENEMAS_AND_SUPPOSITORIES	162	/*Enemas and Suppositories*/
#define PRESCRIPTION_ANIMAL_CONSUMPTION	163	/*Prescription-Animal Consumption*/
#define NONPRESCRIPTION_ANIMAL_CONSUMPTION	164	/*Nonprescription-Animal Consumption*/
#define NON_PRESC SOLD_TO_HOSPITALS_HUMAN	165	/*Non-Presc Sold to Hospitals-Human*/
#define PRESC SOLD_TO_HOSPITALS_HUMAN	166	/*Presc Sold to Hospitals-Human*/
#define NON_PRESC SOLD_TO_HOSPITALS_ANIMALS	167	/*Non-Presc Sold to Hospitals-Animals*/
#define PRESC SOLD_TO_HOSPITALS_ANIMALS	168	/*Presc Sold to Hospitals-Animals*/
#define TAXABLE AND NONTAXABLE_BUNDLED_TOGETHER	169	/*Taxable and Nontaxable Bundled Together*/
#define FREE_SAMPLE_HUMAN_USE	170	/*Free Sample- Human Use*/
#define FREE_SAMPLE_PRESC_HUMAN_USE	171	/*Free Sample- Presc-Human Use*/
#define FREE_SAMPLE_ANIMAL_USE	172	/*Free Sample- Animal Use*/
#define FREE_SAMPLE_PRESC_ANIMAL_USE	173	/*Free Sample- Presc-Animal Use*/

#define MONITORS_LESS_THAN_4_INCHES	174	/*Monitors
Less Than 4 Inches*/		
#define MONITORS_BETWEEN_5_14_INCHES	175	/*Monitors
Between 5-14 inches*/		
#define MONITORS_BETWEEN_THAN_15_34_INCHES	176	/*Monitors
Between Than 15-34 Inches*/		
#define MONITORS_GREATER_THAN_35_INCHES	177	/*Monitors
Greater Than 35 Inches*/		
#define UNLEADED_FUEL_W_O_EXCISE_TAX	178	/*Unleaded
Fuel w/o Excise Tax*/		
#define DIESEL_FUEL_W_O_EXCISE_TAX	179	/*Diesel Fuel
w/o Excise Tax*/		
#define GASOHOL_W_O_EXCISE_TAX	180	/*Gasohol w/o
Excise Tax*/		
#define FUEL_TO_COMMON_CARRIERS_W_O_EXCISE_TAX	181	/*Fuel to
Common Carriers w/o Excise Tax*/		
#define FUEL_PASSENGER_COMMON_CARRIER_W_O_EXCISE_TAX	182	/*Fuel-
Passenger Common Carrier w/o Excise Tax*/		
#define UNLEADED_FUEL_W_EXCISE_TAX	183	/*Unleaded
Fuel w/Excise Tax*/		
#define DIESEL_FUEL_W_EXCISE_TAX	184	/*Diesel Fuel
w/Excise Tax*/		
#define GASOHOL_W_EXCISE_TAX	185	/*Gasohol
w/Excise Tax*/		
#define FUEL_TO_COMMON_CARRIERS_W_EXCISE_TAX	186	/*Fuel to
Common Carriers w/Excise Tax*/		
#define FUEL_PASSENGER_COMMON_CARRIER_W_EXCISE_TAX	187	/*Fuel-
Passenger Common Carrier w/ Excise Tax*/		
#define FUEL_FOR_OFF_ROAD PURPOSES	188	/*Fuel For Off
Road Purposes*/		
#define JET_FUEL	189	/*Jet Fuel*/
#define JET_FUEL_COMMON_CARRIERS	190	/*Jet Fuel--
Common Carriers*/		
#define APPLIANCES	191	/*Appliances*/
#define BABY_OIL	192	/*Baby Oil*/
#define US_FLAG	193	/*US Flag*/
#define COINS_WORK_OF_ART_PURE_METAL	194	/*Coins-Work
of Art-Pure Metal*/		
#define COINS_FOREIGN_CURRENCY_PURE_METAL	195	/*Coins-
Foreign Currency-Pure Metal*/		
#define COINS_COLLECTIBLE_PURE_METAL	196	/*Coins-
Collectible-Pure Metal*/		
#define COINS_INVESTMENT_PURPOSES_PURE_METAL	197	/*Coins-
Investment Purposes-Pure Metal*/		
#define COINS_WORK_OF_ART_90_PCT_OR_GREATER	198	/*Coins-Work
of Art-90% or Greater*/		
#define COINS_FOREIGN_CURRENCY_90_PCT_OR_GREATER	199	/*Coins-
Foreign Currency-90% or Greater*/		
#define COINS_COLLECTIBLE_90_PCT_OR_GREATER	200	/*Coins-
Collectible-90% or Greater*/		
#define COINS_INVESTMENT_PURPOSES_90_PCT_OR_GREATER	201	/*Coins-
Investment Purposes-90% or Greater*/		
#define COINS_WORK_OF_ART_BETWEEN_80_90_PCT	202	/*Coins-Work
of Art-Between 80-90%*/		
#define COINS_FOREIGN_CURRENCY_BETWEEN_80_90_PCT	203	/*Coins-
Foreign Currency-Between 80-90%*/		
#define COINS_COLLECTIBLE_BETWEEN_80_90_PCT	204	/*Coins-
Collectible-Between 80-90%*/		
#define COINS_INVESTMENT_PURPOSES_BETWEEN_80_90_PCT	205	/*Coins-
Investment Purposes-Between 80-90%*/		
#define COINS_WORK_OF_ART_LESS_THAN_80_PCT	206	/*Coins-Work
of Art-Less Than 80%*/		
#define COINS_FOREIGN_CURRENCY_LESS_THAN_80_PCT	207	/*Coins-
Foreign Currency-Less Than 80%*/		
#define COINS_COLLECTIBLE_LESS_THAN_80_PCT	208	/*Coins-
Collectible-Less Than 80%*/		
#define COINS_INVESTMENT_PURPOSES_LESS_THAN_80_PCT	209	/*Coins-
Investment Purposes-Less Than 80%*/		
#define UNCANCELLED_STAMPS_FOR_COLLECTIBLE PURPOSES	210	/*Uncancelled
Stamps-For Collectible Purposes*/		

```

#define CANCELLED_STAMPS 211 /*Cancelled
Stamps*/
#define UNCANCELLED_STAMPS_FOR_POSTAGE PURPOSES 212 /*Uncancelled
Stamps-For Postage Purposes*/
#define CASKETS_FOR_HUMAN_REMAINS 213 /*Caskets-For
Human Remains*/
#define BURIAL_VAULTS_FOR_HUMAN_REMAINS 214 /*Burial
Vaults-For Human Remains*/
#define CASKETS_FOR_ALL_OTHER_REMAINS 215 /*Caskets-For
All Other Remains*/
#define BURIAL_VAULTS_FOR_ALL_OTHER_REMAINS 216 /*Burial
Vaults-For All Other Remains*/
#define HEADSTONE_BURIAL_MARKER_W_O_INSTALLATION 217 /*Headstone/Burial Marker-w/o Installation*/
#define HEADSTONE_BURIAL_MARKER_W_INSTALLATION 218 /*Headstone/Burial Marker-w/ Installation*/
#define SANITARY_NAPKINS_OR_TAMPONS 219 /*Sanitary
Napkins or Tampons*/
#define OTHER_STATE_FLAG 220 /*Other State
Flag*/
#define HOME_STATE_FLAG 221 /*Home State
Flag*/
#define POW_FLAG 222 /*POW Flag*/
#define GROOMING_AND_HYGIENE_PRODUCTS_FOR_HUMAN_USE 223 /*Grooming and
Hygiene Products for Human Use*/
#define GROOMING_AND_HYGIENE_PRODUCTS_FOR_ANIMAL_USE 224 /*Grooming and
Hygiene Products for Animal Use*/
#define TOOTHPASTE_TOOTHBRUSH_DENTAL_FLOSS 225 /*Toothpaste,
Toothbrush, Dental Floss*/
#define GAMING_COINS_METAL_CONTENT_IS_GREATER_THAN_80_PCT 226 /*Gaming
Coins-Metal Content is Greater than 80%*/
#define GAMING_COINS_METAL_CONTENT_IS_LESS_THAN_80_PCT 227 /*Gaming
Coins-Metal Content is Less than 80%*/
#define MARINE_EQUIPMENT 228 /*Marine
Equipment*/
#define CHARCOAL 229 /*Charcoal*/
#define COUPON_BOOKS 230 /*Coupon
Books*/
#define SUPPLIES_AND_FOOD_FOR_SEEING_EYE_DOG 231 /*Supplies and
Food for Seeing Eye Dog*/
#define NON_LEAD_BASED_BATTERIES 232 /*Non-Lead
Based Batteries*/
#define CANDY 233 /*Candy*/
#define CHEWING_GUM 234 /*Chewing
Gum*/
#define FOOD_ADDITIVES 235 /*Food
Additives*/
#define CONFECTIONARY_ITEMS 236 /*Confectionary Items*/
#define ICE 237 /*Ice*/
#define DIETARY_SUPPLEMENTS_QUALIFY 238 /*Dietary
Supplements-Qualify*/
#define DIETARY_SUPPLEMENTS_NON_QUALIFY 239 /*Dietary
Supplements-Non Qualify*/
#define RETAIL_PUBLISHED_MONTHLY 240 /*Retail -
Published Monthly*/
#define RETAIL_PUBLISHED_ANNUALLY 241 /*Retail -
Published Annually*/
#define RETAIL_PUBLISHED_SEMI_MONTHLY 242 /*Retail -
Published Semi-Monthly*/
#define RETAIL_PUBLISHED_SEMI_ANNUALLY 243 /*Retail -
Published Semi-Annually*/
#define RETAIL_PUBLISHED_QUARTERLY 244 /*Retail -
Published Quarterly*/
#define RETAIL_PUBLISHED_WEEKLY 245 /*Retail -
Published Weekly*/
#define SUBSCRIPTION_MONTHLY_US_MAIL 246 /*Subscription-Monthly-US Mail*/
#define SUBSCRIPTION_ANNUALLY_DELIVERED_BY_US_MAIL 247 /*Subscription-Annually-Delivered by US Mail*/

```

#define SUBSCRIPTION_SEMI_MONTHLY_DELIVERED_BY_US_MAIL	248
/*Subscription-Semi-Monthly-Delivered by US Mail*/	
#define SUBSCRIPTION_SEMI_ANNUALLY_DELIVERED_BY_US_MAIL	249
/*Subscription-Semi-Annually-Delivered by US Mail*/	
#define SUBSCRIPTION_QUARTERLY_DELIVERED_BY_US_MAIL	250
/*Subscription-Quarterly-Delivered by US Mail*/	
#define SUBSCRIPTION_WEEKLY_DELIVERED_BY_US_MAIL	251
/*Subscription-Weekly-Delivered by US Mail*/	
#define SUBSCRIPTION_MONTHLY_NOT_DELIVERED_BY_US_MAIL	252
/*Subscription-Monthly-Not Delivered by US Mail*/	
#define SUBSCRIPTION_ANNUALLY_NOT_DELIVERED_BY_US_MAIL	253
/*Subscription-Annually-Not Delivered by US Mail*/	
#define SUBSCRIPTION_SEMI_MONTHLY_NOT_DELIVERED_BY_US_MAIL	254
/*Subscription-Semi-Monthly-Not Delivered by US Mail*/	
#define SUBSCRIPTION_SEMI_ANNUALLY_NOT_DELIVERED_BY_USMAIL	255
/*Subscription-Semi-Annually-Not Delivered by USMail*/	
#define SUBSCRIPTION_QUARTERLY_NOT_DELIVERED_BY_US_MAIL	256
/*Subscription-Quarterly-Not Delivered by US Mail*/	
#define SUBSCRIPTION_WEEKLY_NOT_DELIVERED_BY_US_MAIL	257
/*Subscription-Weekly-Not Delivered by US Mail*/	
#define SUBSCRIPTION_MONTHLY_DOOR_TO_DOOR_DELIVERY	258
/*Subscription-Monthly-Door to Door Delivery*/	
#define SUBSCRIPTION_ANNUALLY_DOOR_TO_DOOR_DELIVERY	259
/*Subscription-Annually-Door to Door Delivery*/	
#define SUBSCRIPTION_SEMI_MONTHLY_DOOR_TO_DOOR_DELIVERY	260
/*Subscription-Semi-Monthly-Door to Door Delivery*/	
#define SUBSCRIPTION_SEMI_ANNUALLY_DOOR_TO_DOOR_DELIVERY	261
/*Subscription-Semi-Annually-Door to Door Delivery*/	
#define SUBSCRIPTION_QUARTERLY_DOOR_TO_DOOR_DELIVERY	262
/*Subscription-Quarterly-Door to Door Delivery*/	
#define SUBSCRIPTION_WEEKLY_DOOR_TO_DOOR_DELIVERY	263
/*Subscription-Weekly-Door to Door Delivery*/	
#define EQUIPMENT_EXISTING_FACILITIES	264
Existing Facilities*/	/*Equipment-
#define EQUIPMENT_ECONOMIC_EXPANSION_OF_FACILITIES	265
Economic Expansion of Facilities*/	/*Equipment-
#define EQUIPMENT_PHYSICAL_EXPANSION_OF_FACILITIES	266
Physical Expansion of Facilities*/	/*Equipment-
#define EQUIPMENT_NEW_FACILITIES	267
New Facilities*/	/*Equipment-
#define REPAIR_PARTS_EXISTING_FACILITIES	268
parts-Existing Facilities*/	/*Repair
#define REPAIR_PARTS_ECONOMIC_EXPANSION	269
Parts-Economic Expansion*/	/*Repair
#define REPAIR_PARTS_PHYSICAL_EXPANSION	270
Parts-Physical Expansion*/	/*Repair
#define REPAIR_PARTS_NEW_FACILITIES	271
Parts-New Facilities*/	/*Repair
#define REPAIR_LABOR_SEPARATELY_STATED	272
Labor-Separately Stated*/	/*Repair
#define REPAIR_LABOR_NOT_SEPARATELY_STATED	273
Labor-Not Separately Stated*/	/*Repair
#define INSTALLATION_LABOR_EQUIPMENT_SEPARATELY_STATED	274
Labor-Equipment-Separately Stated*/	/*Installation
#define INSTALLATION_LABOR_EQUIPMENT_NOT_SEPARATELY_STATED	275
Labor-Equipment-Not Separately Stated*/	/*Installation
#define AUTOMOBILE_SPECIFIC_EQUIPMENT_SEPARATELY_STATED	276
Specific -Equipment-Separately Stated*/	/*Automobile
#define OUTSIDE_INSTALLATION_LABOR_EQUIP_SEPARATELY_STATED	277
Installation Labor-Equip-Separately Stated*/	/*Outside
#define REPAIR_EQUIPMENT_SEPARATELY_STATED	278
Equipment-Separately Stated*/	/*Repair
#define REPLACEMENT_EQUIPMENT_SEPARATELY_STATED	279
Equipment-Separately Stated*/	/*Replacement
#define CLEAN_ROOM_EQUIPMENT	280
Equipment*/	/*Clean room
#define ENVIRONMENTAL_CONTROL_EQUIP	281
/*Environmental Control Equip*/	
#define SAFETY_EQUIP	282
Equip*/	/*Safety

#define PACKING_AND_SHIPPING_EQUIP	283	/*Packing and
Shipping_Equip*/		
#define INTRAPLANT_EQUIP	284	/*Intraplant
Equip*/		
#define HAND_TOOLS	285	/*Hand Tools*/
#define WAREHOUSE_EQUIPMENT	286	/*Warehouse
Equipment*/		
#define NOT_HOME_USE_WITHOUT_A_PRESCRIPTION	287	/*Not Home
Use-Without A Prescription*/		
#define NOT_HOME_USE_WITH_A_PRESCRIPTION	288	/*Not Home
Use-With A Prescription*/		
#define NOT_HOME_USE_PAID_FOR_BY_MEDICARE	289	/*Not Home
Use-Paid For By Medicare*/		
#define NOT_HOME_USE_REIMBURSED_BY_MEDICARE	290	/*Not Home
Use-Reimbursed By Medicare*/		
#define NOT_HOME_USE_PAID_FOR_BY_MEDICAID	291	/*Not Home
Use-Paid For By Medicaid*/		
#define NOT_HOME_USE_REIMBURSED_BY_MEDICAID	292	/*Not Home
Use-Reimbursed By Medicaid*/		
#define HOME_USE_WITHOUT_A_PRESCRIPTION	293	/*Home Use-
Without A Prescription*/		
#define HOME_USE_WITH_A_PRESCRIPTION	294	/*Home Use-
With A Prescription*/		
#define HOME_USE_PAID_FOR_BY_MEDICARE	295	/*Home Use-
Paid For By Medicare*/		
#define HOME_USE_REIMBURSED_BY_MEDICARE	296	/*Home Use-
Reimbursed By Medicare*/		
#define HOME_USE_PAID_FOR_BY_MEDICAID	297	/*Home Use-
Paid For By Medicaid*/		
#define HOME_USE_REIMBURSED_BY_MEDICAID	298	/*Home Use-
Reimbursed By Medicaid*/		
#define EQUIP_WITHOUT_A_PRESCRIPTION	299	/*Equip
Without a Prescription*/		
#define EQUIP_WITH_A_PRESCRIPTION	300	/*Equip With a
Prescription*/		
#define EQUIP_PAID_FOR_BY_MEDICARE	301	/*Equip Paid
for by Medicare*/		
#define EQUIP_REIMBURSED_BY_MEDICARE	302	/*Equip
Reimbursed by Medicare*/		
#define EQUIP_PAID_FOR_BY_MEDICAID	303	/*Equip Paid
for by Medicaid*/		
#define EQUIP_REIMBURSED_BY_MEDICAID	304	/*Equip
Reimbursed by Medicaid*/		
#define GENERAL_WITHOUT_A_PRESCRIPTION	305	/*General-
Without a Prescription*/		
#define GENERAL_WITH_A_PRESCRIPTION	306	/*General-With
a Prescription*/		
#define GENERAL_PAID_FOR_BY_MEDICARE	307	/*General-Paid
for by Medicare*/		
#define GENERAL_REIMBURSED_BY_MEDICARE	308	/*General-
Reimbursed by Medicare*/		
#define GENERAL_PAID_FOR_BY_MEDICAID	309	/*General-Paid
for by Medicaid*/		
#define GENERAL_REIMBURSED_BY_MEDICAID	310	/*General-
Reimbursed by Medicaid*/		
#define CORRECTIVE_EYEGLASSES_WITHOUT_A_PRESCRIPTION	311	/*Corrective
Eyeglasses-Without a Prescription*/		
#define CORRECTIVE_EYEGLASSES_WITH_A_PRESCRIPTION	312	/*Corrective
Eyeglasses-With a Prescription*/		
#define CORRECTIVE_EYEGLASSES_PAID_FOR_BY_MEDICARE	313	/*Corrective
Eyeglasses-Paid for by Medicare*/		
#define CORRECTIVE_EYEGLASSES_REIMBURSED_BY_MEDICARE	314	/*Corrective
Eyeglasses-Reimbursed by Medicare*/		
#define CORRECTIVE_EYEGLASSES_PAID_FOR_BY_MEDICAID	315	/*Corrective
Eyeglasses-Paid for by Medicaid*/		
#define CORRECTIVE_EYEGLASSES_REIMBURSED_BY_MEDICAID	316	/*Corrective
Eyeglasses-Reimbursed by Medicaid*/		
#define CONTACT_LENSES_WITHOUT_A_PRESCRIPTION	317	/*Contact
Lenses-Without a prescription*/		
#define CONTACT_LENSES_WITH_A_PRESCRIPTION	318	/*Contact
Lenses-With a prescription*/		

#define CONTACT_LENSSES_PAID_FOR_BY_MEDICARE	319	/*Contact
Lenses-Paid for by Medicare*/		
#define CONTACT_LENSSES_REIMBURSED_BY_MEDICARE	320	/*Contact
Lenses-Reimbursed by Medicare*/		
#define CONTACT_LENSSES_PAID_FOR_BY_MEDICAID	321	/*Contact
Lenses-Paid for by Medicaid*/		
#define CONTACT_LENSSES_REIMBURSED_BY_MEDICAID	322	/*Contact
Lenses-Reimbursed by Medicaid*/		
#define HEARING_AIDS_WITHOUT_A_PRESCRIPTION	323	/*Hearing
Aids-Without a Prescription*/		
#define HEARING_AIDS_WITH_A_PRESCRIPTION	324	/*Hearing
Aids-With a Prescription*/		
#define HEARING_AIDS_PAID_FOR_BY_MEDICARE	325	/*Hearing
Aids-Paid for by Medicare*/		
#define HEARING_AIDS_REIMBURSED_BY_MEDICARE	326	/*Hearing
Aids-Reimbursed by Medicare*/		
#define HEARING_AIDS_PAID_FOR_BY_MEDICAID	327	/*Hearing
Aids-Paid for by Medicaid*/		
#define HEARING_AIDS_REIMBURSED_BY_MEDICAID	328	/*Hearing
Aids-Reimbursed by Medicaid*/		
#define DENTAL_PROSTHESIS_WITHOUT_A_PRESCRIPTION	329	/*Dental
Prosthesis-Without a prescription*/		
#define DENTAL_PROSTHESIS_WITH_A_PRESCRIPTION	330	/*Dental
Prosthesis-With a prescription*/		
#define DENTAL_PROSTHESIS_PAID_FOR_BY_MEDICARE	331	/*Dental
Prosthesis-Paid for by Medicare*/		
#define DENTAL_PROSTHESIS_REIMBURSED_BY_MEDICARE	332	/*Dental
Prosthesis-Reimbursed by Medicare*/		
#define DENTAL_PROSTHESIS_PAID_FOR_BY_MEDICAID	333	/*Dental
Prosthesis-Paid for by Medicaid*/		
#define DENTAL_PROSTHESIS_REIMBURSED_BY_MEDICAID	334	/*Dental
Prosthesis-Reimbursed by Medicaid*/		
#define LOW_EMISSION_VEHICLE_EXCEEDING_10000_LBS	335	/*Low Emission
Vehicle exceeding 10,000 lbs*/		
#define VEHICLES SOLD TO NATIVE_AMERICANS	336	/*Vehicles
sold to Native Americans*/		
#define MOTOR_VEHICLES_USING_ALTERNATIVE_FUELS	337	/*Motor
Vehicles using Alternative Fuels*/		
#define LOW_SPEED_ELECTRICAL_VEHICLES	338	/*Low Speed
Electrical Vehicles*/		
#define SALE_TO_NON_RESIDENTS	339	/*Sale to Non-
Residents*/		
#define TRUCKS_GENERAL_RULE	340	/*Trucks -
General Rule*/		
#define USED_IN_INTERSTATE_COMMERC	341	/*Used in
Interstate Commerce*/		
#define TRAILER_13_16_AND_HALF_TONS_INTERSTATE_COMMERC	342	/*Trailer 13-
16.5 Tons-Interstate Commerce*/		
#define TRACTOR_EXCEEDS_16_AND_HALF_TONS_INTERSTATE_COMMERC	343	/*Tractor
Exceeds 16.5 Tons-Interstate Commerce*/		
#define USED_AS_A_CONTACT_CARRIER	344	/*Used as a
Contact Carrier*/		
#define TRAILER_EXCEEDS_13_TONS_CONTRACT_CARRIER	345	/*Trailer
Exceeds 13 Tons-Contract Carrier*/		
#define TRACTOR_EXCEEDS_16_AND_HALF_TONS_CONTRACT_CARRIER	346	/*Tractor
Exceeds 16.5 Tons-Contract Carrier*/		
#define MOTOR_BOATS	347	/*Motor
Boats*/		
#define AIRCRAFT	348	/*Aircraft*/
#define AIRCRAFT_FOR_INTERSTATE_TRANSPORT	349	/*Aircraft for
Interstate Transport*/		
#define BATTERIES_LESS_THAN_12_VOLTS_LEAD_BASED	350	/*Batteries
Less than 12 Volts--Lead Based*/		
#define BATTERIES_GREATER_THAN_12_VOLTS_LEAD_BASED	351	/*Batteries
Greater than 12 Volts--Lead Based*/		
#define MOTORCYCLES	352	
/*Motorcycles*/		
#define MOPEDS	353	/*Mopeds*/
#define OFF_ROAD_VEHICLES	354	/*Off-Road
Vehicles*/		

#define SNOWMOBILES	355	
/*Snowmobiles*/		
#define MOTOR_OIL	356	/*Motor Oil*/
#define ANTIFREEZE	357	/*Antifreeze*/
#define BOAT_MOTOR	358	/*Boat Motor*/
#define RETAIL	359	/*Retail*/
#define CD_ROM_MICROFICHE	360	/*CD ROM
Microfiche*/		
#define DIGITAL_PRODUCT_RETAIL	361	/*Digital
Product - Retail*/		
#define DIGITAL_PRODUCT_SUBSCRIPTION	362	/*Digital
Product - Subscription*/		
#define SUBSCRIPTION_MAIL	363	/*Subscription
Mail*/		
#define SUBSCRIPTION_DELIVERY	364	/*Subscription
Delivery*/		
#define FOOD SOLD_BY_A_FOOD_MANUFACTURER	365	/*Food sold by
a Food Manufacturer*/		
#define FOOD SOLD_IN_AN_UNHEATED_STATE	366	/*Food sold in
an unheated state*/		
#define BAKERY_ITEMS	367	/*Bakery
items*/		
#define EMPLOYEE_MEALS_FULL_PRICE	368	/*Employee
Meals--Full Price*/		
#define EMPLOYEE_MEALS_REDUCED_PRICE	369	/*Employee
Meals--Reduced Price*/		
#define EMPLOYEE_MEALS_FREE_TO_EMPLOYEES	370	/*Employee
Meals--Free to Employees*/		
#define TIPS_VOLUNTARY	371	/*Tips,
Voluntary*/		
#define TIPS_MANDATORY	372	/*Tips,
Mandatory*/		
#define NON_TIP_BASED_SERVICE_CHARGE	373	/*Non-Tip
Based Service Charge*/		
#define UNIFORM_RENTAL_SERVICE	374	/*Uniform
Rental Service*/		
#define AUTOMOTIVE_RENTAL_30_DAYS_OR_LESS	375	/*Automotive
Rental--30 Days or Less*/		
#define AUTOMOTIVE_RENTAL_30_180_DAYS	376	/*Automotive
Rental--30-180 Days*/		
#define AUTOMOTIVE_RENTAL_180_DAYS_TO_1_YEAR	377	/*Automotive
Rental--180 Days to 1 Year*/		
#define AUTOMOTIVE_RENTAL_1_YEAR_OR_GREATER	378	/*Automotive
Rental--1 Year or Greater*/		
#define AUTOMOTIVE_RENTAL_0_30_DAYS_TAX_PAID	379	/*Automotive
Rental-0-30 Days-Tax Paid*/		
#define AUTOMOTIVE_RENTAL_31_180_DAYS_TAX_PAID	380	/*Automotive
Rental-31-180 Days-Tax Paid*/		
#define AUTOMOTIVE_RENTAL_180_DAYS_1_YEAR_TAX_PAID	381	/*Automotive
Rental-180 Days-1 Year-Tax Paid*/		
#define AUTOMOTIVE_RENTAL_GREATER_THAN_1_YEAR_TAX_PAID	382	/*Automotive
Rental-Greater than 1 Year-Tax Paid*/		
#define LEASE_OF_AUTOMOBILE_TO_BE_REGISTERED_BY_LESSEE	383	/*Lease of
Automobile to be registered by lessee*/		
#define AMUSEMENT RELATED PROPERTY	384	/*Amusement
Related Property*/		
#define RENTALS INCIDENTAL_TO_SERVICE	385	/*Rentals
Incidental to Service*/		
#define MOVIE_RENTALS_PRIVATE_USE_PHYSICAL_MEDIUM	386	/*Movie
Rentals--Private Use--Physical Medium*/		
#define MOVIE_RENTALS_PRIVATE_USE_DIGITAL_DOWNLOAD	387	/*Movie
Rentals--Private Use--Digital Download*/		
#define MOVIE_RENTALS_AS_PART_OF_EXHIBITION_TO_PUBLIC	388	/*Movie
Rentals--As part of exhibition to public*/		
#define MOVIE_RENTALS_TO_A_TELEVISION_STATION	389	/*Movie
Rentals--To a Television Station*/		
#define DRY_CLEANING_CLOTHING	390	/*Dry
Cleaning-Clothing*/		
#define DRY_CLEANING_NON_CLOTHING	391	/*Dry
Cleaning-Non Clothing*/		

#define CLEANING_SERVICES	392	/*Cleaning Services*/
#define LAUNDRY_AND_CLOTHING_CARE_OTHER_ITEMS	393	/*Laundry and Clothing Care-Other Items*/
#define LAUNDRY_AND_CLOTHING_CARE_CLOTH_DIAPERS	394	/*Laundry and Clothing Care-Cloth Diapers*/
#define LAUNDRY_AND_CLOTHING_CARE_COIN_OPERATED	395	/*Laundry and Clothing Care-Coin Operated*/
#define RUG_CLEANING_OFF_CUSTOMER_PREMISES	396	/*Rug Cleaning off customer premises*/
#define RUG_CLEANING_ON_CUSTOMER_PREMISES	397	/*Rug Cleaning on customer premises*/
#define WASHING_MOTOR_VEHICLES	398	/*Washing Motor Vehicles*/
#define WASHING_MOTOR_VEHICLES_COIN_OPERATED_DEVICE	399	/*Washing Motor Vehicles-Coin Operated Device*/
#define SOLID_WASTE_DISPOSAL	400	/*Solid Waste Disposal*/
#define SWIMMING_POOL_SERVICES	401	/*Swimming Pool Services*/
#define HOTEL_ROOMS_LESS_THAN_28_DAYS	402	/*Hotel Rooms Less than 28 Days*/
#define HOTEL_ROOMS_28_29_DAYS	403	/*Hotel Rooms 28-29 Days*/
#define HOTEL_ROOMS_30_31_DAYS	404	/*Hotel Rooms 30-31 Days*/
#define HOTEL_ROOMS_32_60_DAYS	405	/*Hotel Rooms 32-60 Days*/
#define HOTEL_ROOMS_61_90_DAYS	406	/*Hotel Rooms 61-90 Days*/
#define HOTEL_ROOMS_91_120_DAYS	407	/*Hotel Rooms 91-120 Days*/
#define HOTEL_ROOMS_121_180_DAYS	408	/*Hotel Rooms 121-180 Days*/
#define HOTEL_ROOMS_GREATER_THAN_180_DAYS	409	/*Hotel Rooms Greater than 180 Days*/
#define PHOTOGRAPHY_SERVICES_LABOR_ONLY	410	/*Photography Services-Labor Only*/
#define PHOTOGRAPHY_SERVICES_LABOR_WITH_PICTURES	411	/*Photography Services-Labor with pictures*/
#define NEGATIVE_TO_STANDARD_SIZED_PICTURESLABOR_ONLY	412	/*Negative to Standard Sized Pictures-Labor Only*/
#define NEGATIVE_TO_ENLARGEMENT_SIZED_PICTURESLABOR_ONLY	413	/*Negative to Enlargement Sized Pictures-Labor Only*/
#define PRINTING_SERVICES	414	/*Printing Services*/
#define COPYING_SERVICES	415	/*Copying Services*/
#define CREDIT_CARD_PROCESSING_FEE_PART_OF_SALE	416	/*Credit Card Processing Fee-Part of Sale*/
#define CREDIT_CARD_PROCESSING_FEE_SEPARATE_SALE	417	/*Credit Card Processing Fee-Separate Sale*/
#define PROFESSIONAL_SERVICES	418	/*Professional Services*/
#define INVESTIGATIVE_SERVICES	419	/*Investigative Services*/
#define PROTECTION_AND_SECURITY_SERVICES	420	/*Protection and Security Services*/
#define PEST_CONTROL	421	/*Pest Control*/
#define INTERIOR DESIGN SERVICES	422	/*Interior Design Services*/
#define SKIN_ALTERATION_AND_CARE	423	/*Skin Alteration and Care*/
#define MANICURE_AND_PEDICURE	424	/*Manicure and Pedicure*/
#define FUNERAL_SERVICES	425	/*Funeral Services*/
#define COSMETIC_MEDICAL_PROCEDURES	426	/*Cosmetic Medical Procedures*/

#define PROFESSIONAL_MEDICAL_SERVICES	427	/*Professional
Medical Services*/		
#define TITLE	428	/*Title*/
#define DATING_SERVICE	429	/*Dating
Service*/		
#define ESCROW	430	/*Escrow*/
#define MASSAGES	431	/*Massages*/
#define GIFT_WRAPPING	432	/*Gift
Wrapping*/		
#define FLORAL_SERVICES	433	/*Floral
Services*/		
#define ADVERTISING	434	/*Advertising*/
/*Advertising*/		
#define CREDIT_AND_REPORTING_SERVICES	435	/*Credit and
Reporting Services*/		
#define PERSONNEL_SERVICES	436	/*Personnel
Services*/		
#define LETTERING_SERVICES	437	/*Lettering
Services*/		
#define COLLECTION_SERVICES	438	/*Collection
Services*/		
#define BACKGROUND_MUSIC_SERVICES	439	/*Background
Music Services*/		
#define LOCKSMITH_SERVICES	440	/*Locksmith
Services*/		
#define LOBBYING	441	/*Lobbying*/
#define FLIGHT_INSTRUCTION	442	/*Flight
Instruction*/		
#define INVESTMENT_COUNSELING	443	/*Investment
Counseling*/		
#define SERVICE_CHG_FINANCIAL_INSTITUTION	444	/*Service Chg.
Financial Institution*/		
#define TOW_SERVICE	445	/*Tow
Service*/		
#define TAXIDERMY	446	/*Taxidermy*/
#define TELEPHONE_ANSWERING_SERVICE	447	/*Telephone
Answering Service*/		
#define LIMOUSINE_SERVICE	448	/*Limousine
Service*/		
#define ARCHITECTURE	449	/*Architecture*/
/*Architecture*/		
#define TANNING_SERVICES	450	/*Tanning
Services*/		
#define PET_GROOMING_NOT_DONE_IN_MEDICAL_SETTING	451	/*Pet
Grooming--Not done in medical setting*/		
#define PET_GROOMING_DONE_IN_MEDICAL_SETTING	452	/*Pet
Grooming--Done in Medical Setting*/		
#define ENGRAVING	453	/*Engraving*/
#define HOUSE_MOVING	454	/*House
Moving*/		
#define COUNSELING	455	/*Counseling*/
#define DAY_CARE_SERVICES	456	/*Day Care
Services*/		
#define INVESTMENT_COMMISONS	457	/*Investment
Commissions*/		
#define SALE_OF_INSURANCE	458	/*Sale of
Insurance*/		
#define SPORTING_EVENT_WITH_FOOD_OR_PROPERTY	459	/*Sporting
Event with Food or Property*/		
#define SPORTING_EVENT_WITHOUT_FOOD_OR_PROPERTY	460	/*Sporting
Event without Food or Property*/		
#define AMUSEMENT_PARK_ENTRY_WITH_FOOD_OR_PROPERTY	461	/*Amusement
Park Entry with Food or Property*/		
#define AMUSEMENT_PARK_ENTRY_WITHOUT_FOOD_OR_PROPERTY	462	/*Amusement
Park Entry without Food or Property*/		
#define OTHER_AMUSEMENT_ENTRY_WITH_FOOD_OR_PROPERTY	463	/*Other
Amusement Entry with Food or Property*/		
#define OTHER_AMUSEMENT_ENTRY_WITHOUT_FOOD_OR_PROPERTY	464	/*Other
Amusement Entry without Food or Property*/		
#define ADMISSION_TO_BOXING_AND_WRESTLING	465	/*Admission to
Boxing and Wrestling*/		

```

#define ADMISSION_TO_HORSE_RACING 466 /*Admission to
Horse Racing*/
#define ADMISSION_TO_A_MOTION_PICTURE 467 /*Admission to
a Motion Picture*/
#define TOURS_FOR_8_25_PEOPLE 468 /*Tours for 8-
25 people*/
#define TOURS_FOR_MORE_THAN_25_PEOPLE 469 /*Tours for
more than 25 people*/
#define TOURS_FOR_8_25_PEOPLE_COMMISI 470 /*Tours for 8-
25 people--Commissions*/
#define TOURS_FOR_MORE_THAN_25_PEOPLE_COMMISI 471 /*Tours for
more than 25 people--Commissions*/
#define SPORT_FITNESS_AND_RECREATION_CLUB 472 /*Sport,
Fitness and Recreation Club*/
#define REPAIR_PARTS_GENERAL_RULE 473 /*Repair Parts
- General Rule*/
#define REPAIR_AND_LABOR_COMBINED 474 /*Repair and
Labor Combined*/
#define PARTS_USED_IN_CLOTHING_OR_SHOES 475 /*Parts used
in Clothing or Shoes*/
#define PARTS_USED_IN_REPAIR_OF_COMMERCIAL_AIRPLANES 476 /*Parts used
in repair of Commercial Airplanes*/
#define EXTENDED_WARRANTY_REPAIR_PARTS_USED 477 /*Extended
Warranty - Repair Parts Used*/
#define EXTENDED_WARRANTY_SERVICE_REPAIRS_USED 478 /*Extended
Warranty - Service Repairs Used*/
#define REPAIR_LABOR_GENERAL_RULE 479 /*Repair Labor
- General Rule*/
#define REPAIRS_OF_CLOTHING 480 /*Repairs of
Clothing*/
#define REPAIR_OF_RAILROAD_ROLLING_STOCK_AND_ENGINES 481 /*Repair of
Railroad Rolling Stock and Engines*/
#define REPAIR_OF_MOTOR_VEHICLE 482 /*Repair of
Motor Vehicle*/
#define SHOE_REPAIR_AND_CLEANING 483 /*Shoe Repair
and Cleaning*/
#define AUTOMOTIVE_PAINTING 484 /*Automotive
Painting*/
#define LANDSCAPING_AND_LAWN_CARE 485 /*Landscaping
and Lawn Care*/
#define SNOW_REMOVAL 486 /*Snow
Removal*/
#define REPAIR_OF_COMMERCIAL_JET_AIRCRAFT 487 /*Repair of
Commercial Jet Aircraft*/
#define JEWELRY_REPAIR 488 /*Jewelry
Repair*/
#define STORAGE 489 /*Storage*/
#define PARKING 490 /*Parking*/
#define CANNED_SOFTWARE 491 /*Canned
Software*/
#define MODIFIED_CHARGES 492 /*Modified
Charges*/
#define MODIFIED_SOFTWARE 493 /*Modified
Software*/
#define CUSTOM_SOFTWARE 494 /*Custom
Software*/
#define CANNED_SOFTWARE_LOAD_AND_LEAVE 495 /*Canned
Software-Load and Leave*/
#define CUSTOM_SOFTWARE_LOAD_AND_LEAVE 496 /*Custom
Software-Load and Leave*/
#define LICENSED_SOFTWARE_LOAD_AND_LEAVE 497 /*Licensed
Software-Load and Leave*/
#define MODIFIED_SOFTWARE_LOAD_AND_LEAVE 498 /*Modified
Software-Load and Leave*/
#define DOWNLOADED_CUSTOM_SOFTWARE 499 /*Downloaded
Custom Software*/
#define DOWNLOADED_CANNED_SOFTWARE 500 /*Downloaded
Canned Software*/
#define LICENSED_SOFTWARE_DOWNLOAD 501 /*Licensed
Software-Download*/

```

#define MODIFIED_SOFTWARE_DOWNLOAD	502	/*Modified
Software-Download*/		
#define SOFTWARE_SET_UP_OPTIONAL_CANNED	503	/*Software Set
Up-Optional-Canned*/		
#define SOFTWARE_SET_UP_OPTIONAL_CUSTOM	504	/*Software Set
Up-Optional-Custom*/		
#define SOFTWARE_SET_UP_OPTIONAL_DOWNLOADED	505	/*Software Set
Up-Optional-Downloaded*/		
#define SOFTWARE_SET_UP_OPTIONAL_LOAD_AND_LEAVE	506	/*Software Set
Up-Optional-Load_and_Leave*/		
#define SOFTWARE_SET_UP_OPTIONAL_MODIFIED	507	/*Software Set
Up-Optional-Modified*/		
#define SOFTWARE_SET_UP_MANDATORY_CANNED	508	/*Software Set
Up-Mandatory-Canned*/		
#define SOFTWARE_SET_UP_MANDATORY_CUSTOM	509	/*Software Set
Up-Mandatory-Custom*/		
#define SOFTWARE_SET_UP_MANDATORY_DOWNLOADED	510	/*Software Set
Up-Mandatory-Downloaded*/		
#define SOFTWARE_SET_UP_MANDATORY_LOAD_AND_LEAVE	511	/*Software Set
Up-Mandatory-Load_and_Leave*/		
#define SOFTWARE_SET_UP_MANDATORY_MODIFIED	512	/*Software Set
Up-Mandatory-Modified*/		
#define COMPUTER_CONSULTING OPTIONAL_CANNED	513	/*Computer
Consulting-Optional-Canned*/		
#define COMPUTER_CONSULTING MANDATORY_CANNED	514	/*Computer
Consulting-Mandatory-Canned*/		
#define COMPUTER_CONSULTING OPTIONAL_CUSTOM	515	/*Computer
Consulting-Optional-Custom*/		
#define COMPUTER_CONSULTING MANDATORY_CUSTOM	516	/*Computer
Consulting-Mandatory-Custom*/		
#define COMPUTER_CONSULTING OPTIONAL_DOWNLOADED	517	/*Computer
Consulting-Optional-Downloaded*/		
#define COMPUTER_CONSULTING MANDATORY_DOWNLOADED	518	/*Computer
Consulting-Mandatory-Downloaded*/		
#define COMPUTER_CONSULTING OPTIONAL_LOAD_AND_LEAVE	519	/*Computer
Consulting-Optional-Load_and_Leave*/		
#define COMPUTER_CONSULTING MANDATORY_LOAD_AND_LEAVE	520	/*Computer
Consulting-Mandatory-Load_and_Leave*/		
#define COMPUTER_CONSULTING OPTIONAL_MODIFIED	521	/*Computer
Consulting-Optional-Modified*/		
#define COMPUTER_CONSULTING MANDATORY_MODIFIED	522	/*Computer
Consulting-Mandatory-Modified*/		
#define COMPUTER_TRAINING OPTIONAL_CANNED	523	/*Computer
Training-Optional-Canned*/		
#define COMPUTER_TRAINING MANDATORY_CANNED	524	/*Computer
Training-Mandatory-Canned*/		
#define COMPUTER_TRAINING OPTIONAL_CUSTOM	525	/*Computer
Training-Optional-Custom*/		
#define COMPUTER_TRAINING MANDATORY_CUSTOM	526	/*Computer
Training-Mandatory-Custom*/		
#define COMPUTER_TRAINING OPTIONAL_DOWNLOADED	527	/*Computer
Training-Optional-Downloaded*/		
#define COMPUTER_TRAINING MANDATORY_DOWNLOADED	528	/*Computer
Training-Mandatory-Downloaded*/		
#define COMPUTER_TRAINING OPTIONAL_LOAD_AND_LEAVE	529	/*Computer
Training-Optional-Load_and_Leave*/		
#define COMPUTER_TRAINING MANDATORY_LOAD_AND_LEAVE	530	/*Computer
Training-Mandatory-Load_and_Leave*/		
#define COMPUTER_TRAINING OPTIONAL_MODIFIED	531	/*Computer
Training-Optional-Modified*/		
#define COMPUTER_TRAINING MANDATORY_MODIFIED	532	/*Computer
Training-Mandatory-Modified*/		
#define TIMESHARING_OFF_SITE_USE_OF_CPU_GENERAL_RULE	533	/*Timesharing
- Off-Site Use of CPU - General Rule*/		
#define CPU_LOCATED_OUT_OF_STATE	534	/*CPU Located
out_of_State*/		
#define CIGARETTES	535	/*Cigarettes*/
#define USEFUL_LIFE_BETWEEN_12_MONTHS_AND_3_YEARS	536	/*Useful Life
Between 12 Months and 3 years*/		
#define USEFUL_LIFE_BETWEEN_6_12_MONTHS	537	/*Useful Life
Between 6-12 Months*/		

#define USEFUL_LIFE_EXCEEDS_3_YEARS	538	/*Useful Life
Exceeds 3 years*/		
#define USEFUL_LIFE_LESS_THAN_6_MONTHS	539	/*Useful Life
Less Than 6 Months*/		
#define CANDY SOLD FOR .76_CENTS_OR_MORE	540	/*Candy Sold
For \$0.76 or More*/		
#define CANDY SOLD FOR .50_CENTS_OR_LESS	541	/*Candy Sold
For \$0.50 or Less*/		
#define CANDY BETWEEN .51_AND .75_CENTS	542	/*Candy
Between \$0.51 and \$0.75*/		
#define CHEWING_GUM SOLD FOR .76_CENTS_OR_MORE	543	/*Chewing Gum
Sold For \$0.76 or More*/		
#define CHEWING_GUM SOLD FOR .50_CENTS_OR_LESS	544	/*Chewing Gum
Sold For \$0.50 or Less*/		
#define CHEWING_GUM BETWEEN .51_AND .75_CENTS	545	/*Chewing Gum
Between \$0.51 and \$0.75*/		
#define HOT_PREPARED_FOOD	546	/*Hot Prepared Food*/
#define CARBONATED_BEVERAGES FOR .76_CENTS_OR_MORE	547	/*Carbonated Beverages For \$0.76 or More*/
#define CARBONATED_BEVERAGES FOR .51_TO .75_CENTS	548	/*Carbonated Beverages For \$0.51 to \$0.75*/
#define CARBONATED_BEVERAGES FOR .50_CENTS_OR_LESS	549	/*Carbonated Beverages For \$0.50 or Less*/
#define NON_CARBONATED_BEVERAGES	550	/*Non-Carbonated Beverages*/
#define HOT_BEVERAGES	551	/*Hot Beverages*/
#define PUBLIC_PHYSICAL_TRANSMISSION	552	/*Public-Physical Transmission*/
#define PUBLIC_ELECTRONIC_TRANSMISSION	553	/*Public-Electronic Transmission*/
#define PRIVATE_PHYSICAL_TRANSMISSION	554	/*Private-Physical Transmission*/
#define PRIVATE_ELECTRONIC_TRANSMISSION	555	/*Private-Electronic Transmission*/
#define LICENSED_SOFTWARE_PHYSICAL_TRANSMISSION	556	/*Licensed Software-Physical Transmission*/
#define INFORMATION_SYSTEMS_SERVICES	557	/*Information Systems Services*/
#define REPORT_PHYSICAL_TRANSMISSION	558	/*Report-Physical Transmission*/
#define REMOTE_INFORMATION_RETREIVAL	559	/*Remote Information Retreival*/
#define DOWNLOAD_FROM_INTERNET	560	/*Download from Internet*/
#define DOWNLOAD_TO_PHONE	561	/*Download to Phone*/
#define INTERSTATE_LOCAL_LOOP	562	/*local loop*/
#define LEASE_FACILITIES	563	/*Lease-Facilities*/
#define LEASE_NON_FACILITIES	564	/*Lease-Non-Facilities*/
#define DEBIT_WIRELESS	565	/*Debit-Wireless*/
#define PBX_OUTBOUND_CHANNEL	566	/*PBX Outbound Channel*/
#define PBX_OUTBOUND_CHANNEL_BUNDLE	567	/*PBX Outbound Channel Bundle*/
#define CENTRAL_OFFICE_EQUIPMENT_SALES	568	/*Central Office Equipment-Sales*/
#define CENTRAL_OFFICE_EQUIPMENT_USE	569	/*Central Office Equipment-Use*/
#define DIRECTORY_LISTING	570	/*Directory Listing*/
#define RECYCLING	571	/*Recycling*/
#define DIGITAL_DOWNLOAD	572	/*Digital Download*/
#define RESERVED_573	573	/*Reserved_573*/
#define FIXTURE	574	/*Fixture*/

#define CONFERENCE_BRIDGE_INTRASTATE	575	/*Conference
Bridge-Intrastate*/		
#define CONFERENCE_BRIDGE_INTRASTATE_W_DIAL_IN	576	/*Conference
Bridge-Intrastate w Dial In*/		
#define ENHANCED_FEATURES	577	/*Enhanced
Features*/		
#define PBX	578	/*PBX*/
#define PBX_HIGH_CAPACITY	579	/*PBX High
Capacity*/		
#define HIGH_CAPACITY_EXTENSION	580	/*High
Capacity Extension*/		
#define HIGH_CAPACITY_EXTENSION_BUNDLE	581	/*High
Capacity Extension Bundle*/		
#define HIGH_CAPACITY_OUTBOUND_CHANNEL	582	/*High
Capacity Outbound Channel*/		
#define HIGH_CAPACITY_OUTBOUND_CHANNEL_BUNDLE	583	/*High
Capacity Outbound Channel Bundle*/		
#define DIGITAL_CHANNEL_TIER	584	/*Digital
Channel Tier*/		
#define INTERSTATE_MPLS	585	/*Interstate
MPLS*/		
#define INTRASTATE_MPLS	586	/*Intrastate
MPLS*/		
#define CENTREX_OUTBOUND_CHANNEL	587	/*Centrex
Outbound Channel*/		
#define CENTREX_OUTBOUND_CHANNEL_BUNDLE	588	/*Centrex
Outbound Channel Bundle*/		
#define CONFERENCE_BRIDGE_INTERSTATE	589	/*Conference
Bridge-Interstate*/		
#define RESERVED_590	590	/*Reserved_
590*/		
#define ACCESS_CHARGE_NO_CONTRACT	591	/*Access
Charge-No Contract*/		
#define ACCESS_NUMBER_NO_CONTRACT	592	/*Access
Number-No Contract*/		
#define INFO_SVCS_PRIVATE_PHYSICAL_TRANS	593	/*Info Svcs-
Private Physical Trans*/		
#define INFO_SVCS_PRIVATE_ELECTRONIC_TRANS	594	/*Info Svcs-
Private Electronic Trans*/		
#define DOWNLOADED_LICENSED_SOFTWARE	595	/*Downloaded
Licensed Software*/		
#define ACCESS_LOCAL_ONLY_SERVICE	596	/*Access-Local
Only Service*/		
#define INFO_SVCS_PUBLIC_ELECTRONIC_TRANS	597	/*Info Svcs-
Public Electronic Trans*/		
#define INFO_SVCS_PUBLIC_PHYSICAL_TRANS	598	/*Info Svcs-
Public Physical Trans*/		
#define E_MAIL_HOSTING_SERVICE	599	/*E-mail
Hosting Service*/		
#define REAL_PROPERTY_RENTAL	600	/*Real
Property Rental*/		
#define RESERVED_601	601	/*Reserved -
601*/		
#define SVCS_PROFESSIONAL	602	/*Services-
Professional*/		
#define ONLINE_SERVICES	603	/*Online
Services*/		
#define LEASE_FACILITIES_LOCAL_SERVICE	604	/*Lease-
Facilities-Local Service*/		
#define LEASE_NON_FACILITIES_LOCAL_SVC	605	/*Lease-Non-
Facilities-Local Svc*/		
#define RESERVED_606	606	/*Reserved _
606*/		
#define RESERVED_607	607	/*Reserved_
607*/		
#define CONFERENCE_BRIDGE_INTERSTATE_W_DIAL_IN	608	/*Conference
Bridge Interstate w Dial In*/		
#define STREAMING_VIDEO	609	/*Streaming
Video*/		
#define EARLY_TERMINATION_FEES	610	/*Early
Termination Fees*/		

#define RESERVED_611	611	/*Reserved-
611*/		
#define FCC_SUBSCRIBER_LINE_FEE_MULTI_LINE	612	/*FCC
Subscriber Line Fee Multi line*/		
#define FCC_SUBSCRIBER_LINE_FEE_MULTI_LINE_BUNDLE	613	/*FCC
Subscriber Line Fee Multi Line Bundle*/		
#define TELECOM_EQUIPMENT_RENTAL	614	/*Telecom
Equipment Rental*/		
#define EQUIPMENT_SALES	615	/*Equipment
Sales*/		
#define HURRICANE_SUPPLIES	616	/*Hurricane
Supplies*/		
#define SCHOOL_SUPPLIES	617	/*School
Supplies*/		
#define APPLIANCES_ENERGY_STAR	618	/*Appliances-
Energy Star*/		
#define HUNTING_SUPPLIES	619	/*Hunting
Supplies*/		
#define TAKE_AND_BAKE_PIZZA	620	/*Take and
Bake Pizza*/		
#define BULK_ITEMS	621	/*Bulk Items*/
#define TEXT_MESSAGE	622	/*Text
Message*/		
#define CENTREX_INVOICE	623	/*Centrex
Invoice*/		
#define WIRELESS_SERVICE	624	/*Wireless
Service*/		
#define CUSTOMER_PREMISE_EQUIP_RENTAL	625	/*Customer
Premise Equip Rental*/		
#define CUSTOMER_PREMISE_EQUIP_RENTAL	625	/*Customer
Premise Equip Rental*/		
#define RESERVED_626_	626	/*Reserved-626
*/		
#define INTERNET_ACCESS	627	/*Internet
Access*/		
#define RESERVED_628_	628	/*Reserved-628
*/		
#define MESSAGING_SERVICES	629	/*Messaging
Services*/		
#define PRIVATE_LINE_10_PCT_RULE	630	/*Private Line
(10% Rule)*/		
#define DATA_10_PCT_RULE	631	/*Data (10%
Rule)*/		
#define SERVICE_CONTRACTS	632	/*Service
Contracts*/		
#define DOWNLOADED_VIDEO	633	/*Downloaded
Video*/		
#define SERV_TYPE_RESERVED_634	634	/*Reserved*/
#define TOLL_FREE_NUMBER	635	/*Toll-Free
Number*/		
#define REMOTELY_ACCESSED_SOFTWARE	636	/*Remotely
Accessed Software*/		
#define REMOTE_ACCESS_OF_SOFTWARE	637	/*Remote
Access of Software*/		
#define SECURITY_MONITORING_SERVICES	638	/*Security
Monitoring Services*/		
#define STREAMING_INTERNET_VIDEO	639	/*Streaming
Internet Video*/		
#define RESTOCKING_FEE	640	/*Restocking
Fee*/		
#define FCC_SUBSCRIBER_LINE_CHARGE_CENTREX	641	/*FCC
Subscriber Line Charge Centrex*/		
#define FCC_SUBSCRIBER_LINE_CHARGE_CENTREX_BUNDLE	642	/*FCC
Subscriber Line Charge Centrex Bundle*/		
#define DEBIT_WIRELESS_INDIRECT_NON_CARRIER_SALE	643	/*Debit-
Wireless (Indirect Non-Carrier Sale)*/		
#define INFO_SVCS_PUB_ELEC_TRANS_FIN_AND_SECURITIES	644	/*Info Svcs-
Pub Elec Trans (Fin and Securities)*/		
#define PUBLIC_ELEC_TRANS_FINANCIAL_AND_SECURITIES	645	/*Public-Elec
Trans (Financial and Securities)*/		

#define INFORMATION_RETRIEVAL_PROVIDER_DATA	646	/*Information Retrieval (Provider Data)*/
#define REMOTE_INFORMATION_RETRIEVAL_PROVIDER_DATA	647	/*Remote Information Retrieval (Provider Data)*/
#define INTRASTATE_WITH_FCC_JURISDICTION	648	/*Intrastate with FCC Jurisdiction*/
#define INTERSTATE_WITHOUT_FCC_JURISDICTION	649	/*Interstate without FCC Jurisdiction*/
#define MPLS_INTRASTATE_ACTIVATION	650	/*MPLS Intrastate Activation*/
#define MPLS_INSTALL	651	/*MPLS Install*/
#define MPLS_SERVICE	652	/*MPLS Service*/
#define MPLS_INTERSTATE_ACTIVATION	653	/*MPLS Interstate Activation*/
#define EQUIPMENT_RENTAL_BASIC	654	/*Equipment Rental Basic*/
#define LOCKED_CELL_PHONE	655	/*Locked Cell Phone*/
#define SERVICE_INDIA_INTERSTATE_SUPPLY	656	/*Service - India Interstate Supply*/
#define SERVICE_INDIA_INTRASTATE_SUPPLY	657	/*Service - India Intrastate Supply*/
#define PRODUCT_INDIA_INTERSTATE_SUPPLY	658	/*Product - India Interstate Supply*/
#define PRODUCT_INDIA_INTRASTATE_SUPPLY	659	/*Product - India Intrastate Supply*/

```
/* Deprecated constants */
#define N800
#define LOCAL_LOOP
#define CABLE_TELEVISION_BASIC_SERVICE
#define LOCAL_ACTIVATION
#define LOCAL_INSTALL
#define DIR_AD
#define NUMBER_PORTABILITY_RECOVERY
#define CENTREX_EXTENSION
#define TRUNK
#define LICENSE_SOFTWARE
#define BEVERAGE_ABOVE_7_PCTCONTENT_BY_WEIGHT
BEVERAGE_ABOVE_7_PCT_CONTENT_BY_WEIGHT_
#define EDUCATIONAL_COLLEGE_OR_TRADE_SCHOOL
#define TOOTHPASTE
#define LOW_EMISSION_VEHICLE_EXCEEDING_10
LOW_EMISSION_VEHICLE_EXCEEDING_10000_LBS
#define TIPS
#define SPORT
#define INFORMATION_SYSTEM_SERVICES
#define LEASE_FACILITIES_LOCAL_SVC
#define CONFERENCE_BRIDGE_INTERSTATE_W_DIALIN
CONFERENCE_BRIDGE_INTERSTATE_W_DIAL_IN

#endif __cplusplus
}

#endif /* _inc_EZTaxServTypeDefine_ */
```

TOLL_FREE	
ACCESS_CHARGE	
ACCESS_CHARGE	
INSTALL	
INSTALL	
DIRECTORY_ADS	
FCC_SUBSCRIBER_LINE_FEE	
CENTREX_DID_EXTENSION	
CENTREX_TRUNK	
LICENSED_SOFTWARE	
EDUCATIONAL_COLLEGE_AND_TRADE_SCHOOL	
TOOTHPASTE_TOOTHBRUSH_DENTAL_FLOSS	
TIPS_VOLUNTARY	
SPORT_FITNESS_AND_RECREATION_CLUB	
INFORMATION_SYSTEMS_SERVICES	
LEASE_FACILITIES_LOCAL_SERVICE	

```

EZTaxStateType.h

#ifndef _inc_EZTaxStateTypeDefine_
#define _inc_EZTaxStateTypeDefine_

#ifndef __cplusplus
extern "C" {
#endif

/* Define state types */
#define NO_STATE 0
#define Alabama 1
#define Alaska 2
#define Arizona 3
#define Arkansas 4
#define California 5
#define Colorado 6
#define Connecticut 7
#define Delaware 8
#define Washington_DC 9
#define D.C.*/ 10
#define Florida 11
#define Georgia 12
#define Hawaii 13
#define Idaho 14
#define Illinois 15
#define Indiana 16
#define Iowa 17
#define Kansas 18
#define Kentucky 19
#define Louisiana 20
#define Maine 21
#define Maryland 22
#define Massachusetts 23
#define Michigan 24
#define Minnesota 25
#define Mississippi 26
#define Missouri 27
#define Montana 28
#define Nebraska 29
#define Nevada 30
#define New_Hampshire 31
#define New_Jersey 32
#define New_Mexico 33
#define New_York 34
#define North_Carolina 35
#define Carolina*/ 36
#define North_Dakota 37
#define Ohio 38
#define Oklahoma 39
#define Oregon 40
#define Pennsylvania 41
#define Rhode_Island 42
#define South_Carolina 43
#define Carolina*/ 44
#define South_Dakota 45
#define Tennessee 46
#define Texas 47
#define Utah 48
#define Vermont 49
#define Virginia 50
#define Washington 51
#define West_Virginia 52
#define Wisconsin 53
#define Wyoming 54
#define Alberta 55
#define Anguar 56

/* */
/*Alabama*/
/*Alaska*/
/*Arizona*/
/*Arkansas*/
/*California*/
/*Colorado*/
/*Connecticut*/
/*Delaware*/
/*Washington, D.C.*/
/*Florida*/
/*Georgia*/
/*Hawaii*/
/*Idaho*/
/*Illinois*/
/*Indiana*/
/*Iowa*/
/*Kansas*/
/*Kentucky*/
/*Louisiana*/
/*Maine*/
/*Maryland*/
/*Massachusetts*/
/*Michigan*/
/*Minnesota*/
/*Mississippi*/
/*Missouri*/
/*Montana*/
/*Nebraska*/
/*Nevada*/
/*New Hampshire*/
/*New Jersey*/
/*New Mexico*/
/*New York*/
/*North*/
/*North Dakota*/
/*Ohio*/
/*Oklahoma*/
/*Oregon*/
/*Pennsylvania*/
/*Rhode Island*/
/*South*/
/*South Dakota*/
/*Tennessee*/
/*Texas*/
/*Utah*/
/*Vermont*/
/*Virginia*/
/*Washington*/
/*West Virginia*/
/*Wisconsin*/
/*Wyoming*/
/*Alberta*/
/*Anguar*/

```

#define British_Columbia	54	/*British Columbia*/
#define Guam	55	/*Guam*/
#define Koror	56	/*Koror*/
#define Kosrae	57	/*Kosrae*/
#define Manitoba	58	/*Manitoba*/
#define Manua_Islands	59	/*Manua Islands*/
#define New_Brunswick	60	/*New Brunswick*/
#define Newfoundland	61	/*Newfoundland*/
#define Northwest_Territories	62	/*Northwest Territories*/
#define Nova_Scotia	63	/*Nova Scotia*/
#define Ontario	64	/*Ontario*/
#define Peleliu	65	/*Peleliu*/
#define Pohnpei	66	/*Pohnpei*/
#define Prince_Edward_Island	67	/*Prince Edward Island*/
#define Puerto_Rico	68	/*Puerto Rico*/
#define Quebec	69	/*Quebec*/
#define Ralik_Chain	70	/*Ralik Chain*/
#define Ratak_Chain	71	/*Ratak Chain*/
#define Rose_Island	72	/*Rose Island*/
#define Rota	73	/*Rota*/
#define Saint_Croix	74	/*Saint Croix*/
#define Saint_John	75	/*Saint John*/
#define Saint_Thomas	76	/*Saint Thomas*/
#define Saipan	77	/*Saipan*/
#define Saskatchewan	78	/*Saskatchewan*/
#define Swains_Island	79	/*Swains Island*/
#define Tinian	80	/*Tinian*/
#define Truk	81	/*Truk*/
#define Tutulia_Island	82	/*Tutulia Island*/
#define Yap	83	/*Yap*/
#define Yukon_Territory	84	/*Yukon Territory*/
#define Dodecanese	85	/*Dodecanese*/
#define Lesvos	86	/*Lesvos*/
#define Chios	87	/*Chios*/
#define Samos	88	/*Samos*/
#define Thassos	89	/*Thassos*/
#define Samothraki	90	/*Samothraki*/
#define Cyclades	91	/*Cyclades*/
#define North_Sporades	92	/*North Sporades*/
#define Skyros	93	/*Skyros*/
#define Nunavut	94	/*Nunavut*/
#define Bermuda	95	/*Bermuda*/
#define American_Samoa	96	/*American Samoa*/
#define Fed_St_of_Micronesia	97	/*Fed St of Micronesia*/
#define Marshall_Islands	98	/*Marshall Islands*/
#define Northern_Mariana_Islands	99	/*Northern Mariana Islands*/
#define Palau	100	/*Palau*/
#define US_Virgin_Islands	101	/*US Virgin Islands*/
#define Sao_Paulo	102	/*Sao Paulo*/
#define Rio_de_Janeiro	103	/*Rio de Janeiro*/
#define Distrito_Federal	104	/*Distrito Federal*/
#define Sergipe	105	/*Sergipe*/
#define Bahia	106	/*Bahia*/
#define Amapa	107	/*Amapa*/
#define Acre	108	/*Acre*/
#define Alagoas	109	/*Alagoas*/
#define Amazonas	110	/*Amazonas*/
#define Ceara	111	/*Ceara*/

#define Espirito_Santo	112	/*Espírito Santo*/
Santo*/		
#define Goias	113	/*Goias*/
#define Maranhao	114	/*Maranhão*/
#define Mato_Grosso	115	/*Mato Grosso*/
#define Mato_Grosso_do_Sul	116	/*Mato Grosso do Sul*/
#define Minas_Gerais	117	/*Minas Gerais*/
#define Para	118	/*Para*/
#define Paraiba	119	/*Paraíba*/
#define Parana	120	/*Paraná*/
#define Pernambuco	121	/*Pernambuco*/
#define Piaui	122	/*Piauí*/
#define Rio_Grande_do_Norte	123	/*Rio Grande do Norte*/
Norte*/		
#define Rio_Grande_do_Sul	124	/*Rio Grande do Sul*/
Sul*/		
#define Rondonia	125	/*Rondônia*/
#define Roraima	126	/*Roraima*/
#define Santa_Catarina	127	/*Santa Catarina*/
Catarina*/		
#define Tocantins	128	/*Tocantins*/
#define Andhra_Pradesh	129	/*Andhra Pradesh*/
Pradesh*/		
#define Arunachal_Pradesh	130	/*Arunachal Pradesh*/
Pradesh*/		
#define Assam	131	/*Assam*/
#define Bihar	132	/*Bihar*/
#define Chhattisgarh	133	/*Chhattisgarh*/
#define Goa	134	/*Goa*/
#define Gujarat	135	/*Gujarat*/
#define Haryana	136	/*Haryana*/
#define Himachal_Pradesh	137	/*Himachal Pradesh*/
Pradesh*/		
#define Jammu_and_Kashmir	138	/*Jammu and Kashmir*/
Kashmir*/		
#define Jharkhand	139	/*Jharkhand*/
#define Karnataka	140	/*Karnataka*/
#define Kerala	141	/*Kerala*/
#define Madhya_Pradesh	142	/*Madhya Pradesh*/
Pradesh*/		
#define Maharashtra	143	/*Maharashtra*/
#define Manipur	144	/*Manipur*/
#define Meghalaya	145	/*Meghalaya*/
#define Mizoram	146	/*Mizoram*/
#define Nagaland	147	/*Nagaland*/
#define Odisha	148	/*Odisha*/
#define Punjab	149	/*Punjab*/
#define Rajasthan	150	/*Rajasthan*/
#define Sikkim	151	/*Sikkim*/
#define Tamil_Nadu	152	/*Tamil Nadu*/
#define Telangana	153	/*Telangana*/
#define Tripura	154	/*Tripura*/
#define Uttar_Pradesh	155	/*Uttar Pradesh*/
#define Uttarakhand	156	/*Uttarakhand*/
#define West_Bengal	157	/*West Bengal*/
#define Andaman_and_Nicobar	158	/*Andaman and Nicobar*/
Nicobar*/		
#define Chandigarh	159	/*Chandigarh*/
#define Dadra_Nagar_Haveli	160	/*Dadra Nagar Haveli*/
Haveli*/		
#define Daman_and_Diu	161	/*Daman and Diu*/
#define Delhi	162	/*Delhi*/
#define Lakshadweep	163	/*Lakshadweep*/
#define Puducherry	164	/*Puducherry*/

/* Deprecated state type constants */
#define Distric_of_Columbia

Washington_DC

```
#ifdef __cplusplus
}
#endif /* _inc_EZTaxStateTypeDefine_ */
```

EZTaxCountryType.h

```
#ifndef _inc_EZTaxCountryTypeDefine_
#define _inc_EZTaxCountryTypeDefine_

#ifndef __cplusplus
extern "C" {
#endif

/* Foreign countries apply to Telecom transactions only */
/* Define Country Ids */
#define UNITED_STATES_OF_AMERICA 0 /*United States
Of America*/
#define ARGENTINA 2 /*Argentina*/
#define AUSTRALIA 3 /*Australia*/
#define AUSTRIA 4 /*Austria*/
#define AZERBAIJAN 5 /*Azerbaijan*/
#define BARBADOS 6 /*Barbados*/
#define BELGIUM 7 /*Belgium*/
#define BOLIVIA 8 /*Bolivia*/
#define BULGARIA 9 /*Bulgaria*/
#define CAMBODIA 10 /*Cambodia*/
#define CANADA 11 /*Canada*/
#define CHINA 12 /*China*/
#define COLOMBIA 13 /*Colombia*/
#define COSTA_RICA 14 /*Costa Rica*/
#define CROATIA 15 /*Croatia*/
#define CYPRUS 16 /*Cyprus*/
#define CZECH_REPUBLIC 17 /*Czech
Republic*/
#define DENMARK 18 /*Denmark*/
#define ECUADOR 19 /*Ecuador*/
#define ESTONIA 20 /*Estonia*/
#define FIJI 22 /*Fiji*/
#define FINLAND 23 /*Finland*/
#define FRANCE 24 /*France*/
#define GERMANY 25 /*Germany*/
#define GHANA 26 /*Ghana*/
#define ZAMBIA 27 /*Zambia*/
#define HONDURAS 29 /*Honduras*/
#define HUNGARY 30 /*Hungary*/
#define INDONESIA 31 /*Indonesia*/
#define IRELAND 32 /*Ireland*/
#define ITALY 33 /*Italy*/
#define JAPAN 34 /*Japan*/
#define KENYA 35 /*Kenya*/
#define LATVIA 36 /*Latvia*/
#define LITHUANIA 37 /*Lithuania*/
#define LUXEMBOURG 38 /*Luxembourg*/
#define MAURITIUS 40 /*Mauritius*/
#define MEXICO 41 /*Mexico*/
#define MOROCCO 42 /*Morocco*/
#define NAMIBIA 43 /*Namibia*/
#define NETHERLANDS 44 /*Netherlands*/
#define NEW_ZEALAND 45 /*New Zealand*/
#define NORWAY 47 /*Norway*/
#define PANAMA 49 /*Panama*/
#define PAPUA_NEW_GUINEA 50 /*Papua New
Guinea*/
#define PERU 51 /*Peru*/
#define POLAND 52 /*Poland*/
#define PORTUGAL 53 /*Portugal*/
#define ROMANIA 54 /*Romania*/
#define RUSSIA 56 /*Russia*/
#define SINGAPORE 57 /*Singapore*/
#define SLOVENIA 58 /*Slovenia*/
#define SOUTH_AFRICA 59 /*South Africa*/
#define SPAIN 60 /*Spain*/
#define SWEDEN 61 /*Sweden*/
#define SWITZERLAND 62 /*Switzerland*/
```

#define TAIWAN	63	/*Taiwan*/
#define TANZANIA	64	/*Tanzania*/
#define THAILAND	65	/*Thailand*/
#define TRINIDAD_AND_TOBAGO Tobago*/	66	/*Trinidad And
#define TURKEY	67	/*Turkey*/
#define UGANDA	68	/*Uganda*/
#define UKRAINE	69	/*Ukraine*/
#define UNITED_KINGDOM Kingdom*/	70	/*United
#define URUGUAY	71	/*Uruguay*/
#define VENEZUELA	73	/*Venezuela*/
#define VIETNAM	74	/*Vietnam*/
#define GREECE	75	/*Greece*/
#define ICELAND	76	/*Iceland*/
#define INDIA	77	/*India*/
#define NIGERIA	78	/*Nigeria*/
#define SOUTH_KOREA	79	/*South Korea*/
#define SLOVAK_REPUBLIC Republic*/	80	/*Slovak
#define BERMUDA	81	/*Bermuda*/
#define UNSUPPORTED_COUNTRY Country*/	82	/*Unsupported
#define REPUBLIC_OF_MALTA Malta*/	83	/*Republic Of
#define MALAYSIA	84	/*Malaysia*/
#define REUNION_ISLAND Island*/	85	/*Reunion
#define ST_KITTS_AND_NEVIS Nevis*/	86	/*St Kitts And
#define ANTIGUA	87	/*Antigua*/
#define BELIZE	88	/*Belize*/
#define DOMINICA	89	/*Dominica*/
#define DOMINICAN_REPUBLIC Republic*/	90	/*Dominican
#define EL_SALVADOR	91	/*El Salvador*/
#define GRENADE	92	/*Grenada*/
#define GUATEMALA	93	/*Guatemala*/
#define GUYANA	94	/*Guyana*/
#define JAMAICA	95	/*Jamaica*/
#define NICARAGUA	96	/*Nicaragua*/
#define ST_VINCENT	97	/*St Vincent*/
#define CHILE	98	/*Chile*/
#define PHILIPPINES	99	/*Philippines*/
#define ISRAEL	100	/*Israel*/
#define BOSNIA_AND_HERZEGOVINA Herzegovina*/	101	/*Bosnia And
#define LEBANON	102	/*Lebanon*/
#define SERBIA	103	/*Serbia*/
#define EGYPT	104	/*Egypt*/
#define BAHAMAS	105	/*Bahamas*/
#define BRAZIL	106	/*Brazil*/
#define SENEGAL	107	/*Senegal*/
#define BAHRAIN	108	/*Bahrain*/
#define KUWAIT	109	/*Kuwait*/
#define OMAN	110	/*Oman*/
#define QATAR	111	/*Qatar*/
#define SAUDI_ARABIA	112	/*Saudi Arabia*/
#define UNITED_ARAB_EMIRATES Emirates*/	113	/*United Arab
#define USA Of America*/	0	/*United States
*/		
#define ARG	2	/*Argentina*/
#define AUS	3	/*Australia*/
#define AUT	4	/*Austria*/
#define AZE	5	/*Azerbaijan*/
#define BRB	6	/*Barbados*/

```

#define BEL 7 /*Belgium*/
#define BOL 8 /*Bolivia*/
#define BGR 9 /*Bulgaria*/
#define KHM 10 /*Cambodia*/
#define CAN 11 /*Canada*/
#define CHN 12 /*China*/
#define COL 13 /*Colombia*/
#define CRI 14 /*Costa Rica*/
#define HRV 15 /*Croatia*/
#define CYP 16 /*Cyprus*/
#define CZE 17 /*Czech Republic*/
#define DNK 18 /*Denmark*/
#define ECU 19 /*Ecuador*/
#define EST 20 /*Estonia*/
#define FJI 22 /*Fiji*/
#define FIN 23 /*Finland*/
#define FRA 24 /*France*/
#define DEU 25 /*Germany*/
#define GHA 26 /*Ghana*/
#define ZMB 27 /*Zambia*/
#define HND 29 /*Honduras*/
#define HUN 30 /*Hungary*/
#define IDN 31 /*Indonesia*/
#define IRL 32 /*Ireland*/
#define ITA 33 /*Italy*/
#define JPN 34 /*Japan*/
#define KEN 35 /*Kenya*/
#define LVA 36 /*Latvia*/
#define LTU 37 /*Lithuania*/
#define LUX 38 /*Luxembourg*/
#define MUS 40 /*Mauritius*/
#define MEX 41 /*Mexico*/
#define MAR 42 /*Morocco*/
#define NAM 43 /*Namibia*/
#define NLD 44 /*Netherlands*/
#define NZL 45 /*New Zealand*/
#define NOR 47 /*Norway*/
#define PAN 49 /*Panama*/
#define PNG 50 /*Papua New Guinea*/
#define PER 51 /*Peru*/
#define POL 52 /*Poland*/
#define PRT 53 /*Portugal*/
#define ROU 54 /*Romania*/
#define RUS 56 /*Russia*/
#define SGP 57 /*Singapore*/
#define SVN 58 /*Slovenia*/
#define ZAF 59 /*South Africa*/
#define ESP 60 /*Spain*/
#define SWE 61 /*Sweden*/
#define CHE 62 /*Switzerland*/
#define TWN 63 /*Taiwan*/
#define TZA 64 /*Tanzania*/
#define THA 65 /*Thailand*/
#define TTO 66 /*Trinidad And Tobago*/
#define TUR 67 /*Turkey*/
#define UGA 68 /*Uganda*/
#define UKR 69 /*Ukraine*/
#define GBR 70 /*United Kingdom*/
#define URY 71 /*Uruguay*/
#define VEN 73 /*Venezuela*/
#define VNM 74 /*Vietnam*/
#define GRC 75 /*Greece*/
#define ISL 76 /*Iceland*/
#define IND 77 /*India*/
#define NGA 78 /*Nigeria*/
#define KOR 79 /*South Korea*/

```

```

#define SVK 80 /*Slovak
Republic*/
#define BMU 81 /*Bermuda*/
#define UNS 82 /*Unsupported
Country*/
#define MLT 83 /*Republic Of
Malta*/
#define MYS 84 /*Malaysia*/
#define REU 85 /*Reunion
Island*/
#define KNA 86 /*St Kitts And
Nevis*/
#define ATG 87 /*Antigua*/
#define BLZ 88 /*Belize*/
#define DMA 89 /*Dominica*/
#define DOM 90 /*Dominican
Republic*/
#define SLV 91 /*El Salvador*/
#define GRD 92 /*Grenada*/
#define GTM 93 /*Guatemala*/
#define GUY 94 /*Guyana*/
#define JAM 95 /*Jamaica*/
#define NIC 96 /*Nicaragua*/
#define VCT 97 /*St Vincent*/
#define CHL 98 /*Chile*/
#define PHL 99 /*Philippines*/
#define ISR 100 /*Israel*/
#define BIH 101 /*Bosnia And
Herzegovina*/
#define LBN 102 /*Lebanon*/
#define SRB 103 /*Serbia*/
#define EGY 104 /*Egypt*/
#define BHS 105 /*Bahamas*/
#define BRA 106 /*Brazil*/
#define SEN 107 /*Senegal*/
#define BHR 108 /*Bahrain*/
#define KWT 109 /*Kuwait*/
#define OMN 110 /*Oman*/
#define QAT 111 /*Qatar*/
#define SAU 112 /*Saudi Arabia*/
#define ARE 113 /*United Arab
Emirates*/
#define END_OF_COUNTRY_ISO 114 /* This should
always be last numerically */

```

```

/* Deprecated country ids */
#define AMERICAN_SAMOA 1 /*American
Samoa*/
#define ASM 1 /*American
Samoa*/
#define FEDERATED_STATES_OF_MICRONESIA 21 /*Federated
States Of Micronesia*/
#define FSM 21 /*Federated
States Of Micronesia*/
#define GUAM 28 /*Guam*/
#define GUM 28 /*Guam*/
#define MARSHALL_ISLANDS 39 /*Marshall
Islands*/
#define MHL 39 /*Marshall
Islands*/
#define NORTHERN_MARIANA_ISLANDS 46 /*Northern
Mariana Islands*/
#define MNP 46 /*Northern
Mariana Islands*/
#define PALAU 48 /*Palau*/
#define PLW 48 /*Palau*/
#define PUERTO_RICO 55 /*Puerto Rico*/
#define PRI 55 /*Puerto Rico*/

```

```

#define US_VIRGIN_ISLANDS           72          /*Us Virgin
Islands*/
#define VIR_ISLANDS                72          /*Us Virgin

#define AMERICA_SAMOA              AMERICAN_SAMOA
#define NIGERA                      NIGERIA
#define ROM                         ROU
#define BELGIUM                     BELGIUM
#define COLUMBIA                   COLOMBIA

#ifndef __cplusplus
#endif /* _inc_EZTaxCountryTypeDefine_ */

```

8.3 Appendix C EZTaxProto.h

```
#ifndef _inc_EZTaxProto_
#define _inc_EZTaxProto_

/* If defined for C++ environment */

#ifndef __cplusplus
extern "C" {
#endif

#include "EZTaxStruct.h"

/*****
 *  For more information on the API calls please
 *  refer to the EZTax Users Manual
 ****/

/* Initialization / Termination Function Prototypes */
PF_FUNC_SPEC void EZTaxStart(void);
PF_FUNC_SPEC struct taxes_tbl_v98 *EZTaxInitV98(short int tax_log, struct file_path *paths,
                                                 EZTaxSession *session, char *working_dir);
PF_FUNC_SPEC struct enhanced_taxes_tbl *EZTaxInitDirEx(short int tax_log, struct file_path
                                                       *paths, EZTaxSession *session, char *working_dir);
PF_FUNC_SPEC struct enhanced_taxes_tbl *EZTaxInitEx(short int tax_log, struct file_path
                                                    *paths, EZTaxSession *session);
PF_FUNC_SPEC struct enhanced_taxes_tbl *EZTaxInitExMT(short int tax_log, struct file_path
                                                       *paths, EZTaxSession *session, char *working_dir);
PF_FUNC_SPEC struct enhanced_taxes_tbl *EZTaxInitV914(short int tax_log_enabled, struct
                                                       file_path_v914 *paths, EZTaxSession *session);
PF_FUNC_SPEC short int EZTaxExit(void);
PF_FUNC_SPEC short int EZTaxExitSessionEx(EZTaxSession session);

/* Set / Modify Prototypes */
PF_FUNC_SPEC short int EZTaxClearExclusion(EZTaxSession session);
PF_FUNC_SPEC short int EZTaxFreeRates(struct jurisdictionTaxes *taxes);
PF_FUNC_SPEC short int EZTaxGroupResults(EZTaxSession session, unsigned long int groupMode);
PF_FUNC_SPEC short int EZTaxRestoreEx(EZTaxSession session);
PF_FUNC_SPEC short int EZTaxSetLogName(EZTaxSession session, char *fileName);
PF_FUNC_SPEC short int EZTaxSetNexus(EZTaxSession session, struct nexus_table *nexus_tbl, int
                                      nexus_count);
PF_FUNC_SPEC short int EZTaxSetStateExclusion(EZTaxSession session, char country_ISO[4], char
                                              state_abv[3], short int flag);
PF_FUNC_SPEC short int EZTaxSetStateNexus(EZTaxSession session, char country_ISO[4], char
                                         state_abv[3], short int flag);
PF_FUNC_SPEC short int EZTaxSetWorkingDir(char *directory);

/* Jurisdiction Calculation / Conversion Prototypes */
PF_FUNC_SPEC unsigned long int EZTaxCalcJurisdiction(EZTaxSession session, struct J_CodeEx
                                                     *trans, int *err_code);
PF_FUNC_SPEC unsigned long int EZTaxGetJurisdiction(EZTaxSession session, struct J_CodeEx
                                                   *trans, int *err_code);
PF_FUNC_SPEC char* EZTaxPtoFipsEx(EZTaxSession session, unsigned long int PCode, int
                                    *err_code);
PF_FUNC_SPEC unsigned long int EZTaxFtoPCodeEx(EZTaxSession session, char* Fips, int
                                                *err_code);
PF_FUNC_SPEC unsigned long int EZTaxJtoPCodeEx(EZTaxSession session, unsigned long int JCode,
                                               int *err_code);
PF_FUNC_SPEC unsigned long int EZTaxNtoJCodeEx(EZTaxSession session, unsigned long int npanxx,
                                               int *err_code);
PF_FUNC_SPEC unsigned long int EZTaxPtoJCodeEx(EZTaxSession session, unsigned long int PCode,
                                               int *err_code);
PF_FUNC_SPEC unsigned long int EZTaxZtoJCodeEx(EZTaxSession session, struct zip_address_p4
                                               *address, int *err_code);
PF_FUNC_SPEC unsigned long int EZTaxZtoPCodeEx(EZTaxSession session, struct zip_address_p4
                                               *address, int *err_code);
PF_FUNC_SPEC unsigned long int EZTaxCountryToPCode(EZTaxSession session, const char* iso_code,
                                                int *err_code);
```

```

PF_FUNC_SPEC short int EZTaxNextAddressEx(EZTaxSession session, struct address_data_p4
                                         *address, int *err_code);
PF_FUNC_SPEC short int EZTaxJTTType(unsigned long int orig_J_Code, unsigned long int
                                    term_J_Code);
PF_FUNC_SPEC short int EZTaxJTTTypeEx(EZTaxSession session, unsigned long int orig_J_Code,
                                       unsigned long int term_J_Code, int *err_code);
PF_FUNC_SPEC short int EZTaxNTTTypeEx(EZTaxSession session, unsigned long int orig_NPANXX,
                                       unsigned long int term_NPANXX);
PF_FUNC_SPEC short int EZTaxPTTTypeEx(EZTaxSession session, unsigned long int orig_PCode,
                                       unsigned long int term_PCode, int *err_code);
PF_FUNC_SPEC short int EZTaxGetAddressEx(EZTaxSession session, unsigned long int j_code,
                                         struct address_data_p4 *address, int *err_code);
PF_FUNC_SPEC unsigned int EZTaxGetCountryID(EZTaxSession session, unsigned long int PCode, int
                                             *err_code);
PF_FUNC_SPEC unsigned int EZTaxGetStateID(EZTaxSession session, unsigned long int PCode, int
                                           *err_code);

/* Misc Data Retrieval Prototypes */
PF_FUNC_SPEC char *EZTaxGetTaxCatV98(EZTaxSession session, int tax_type);
PF_FUNC_SPEC char *EZTaxGetTaxDescription(EZTaxSession session, int taxCode);
PF_FUNC_SPEC short int EZTaxDbVersion(char* psz_EZTax_dat, char* psz_db_version);
PF_FUNC_SPEC short int EZTaxSessionDbVersion(EZTaxSession session, char* psz_db_version);
PF_FUNC_SPEC short int EZTaxDllVersion(char* psz_library_version);
PF_FUNC_SPEC short int EZTaxGetLogName(EZTaxSession session, char* filename);
PF_FUNC_SPEC short int EZTaxGetRates(EZTaxSession session, unsigned long int pCode, struct
                                     jurisdictionTaxes *taxes, int *err_code);
PF_FUNC_SPEC struct no_tax_tbl *EZTaxGetNoTaxTrans(EZTaxSession session);
PF_FUNC_SPEC short int EZTaxMaxTaxCount(void);
PF_FUNC_SPEC struct rtr_data* EZTaxGetTSR(EZTaxSession session, int *size, short int
                                            returnData);
PF_FUNC_SPEC void EZTaxClearTSR(EZTaxSession session, struct rtr_data* rpt);

/* Customer Mode Related Prototypes */
PF_FUNC_SPEC int EZTaxNextCustomerEx(EZTaxSession session);
PF_FUNC_SPEC struct cust_taxes_tbl_v98 *EZTaxSetCustModeV98(EZTaxSession session, short int
                                                          mode);
PF_FUNC_SPEC struct enhanced_cust_taxes_tbl *EZTaxSetCustModeEx(EZTaxSession session, short
                                                                int mode);

/* Invoice Mode Related Prototypes */
PF_FUNC_SPEC int EZTaxNextInvoiceEx(EZTaxSession session);
PF_FUNC_SPEC struct invoice_taxes_tbl_v98 *EZTaxSetInvoiceModeV98(EZTaxSession session, short
                                                               int mode);
PF_FUNC_SPEC struct enhanced_invoice_taxes_tbl *EZTaxSetInvoiceModeEx(EZTaxSession session,
                                                                     short int mode);

/* Debit Tax Calculation Prototypes*/
PF_FUNC_SPEC short int EZTaxDebitJEx(EZTaxSession session, struct J_CodeEx *trans, int
                                      *err_code);
PF_FUNC_SPEC short int EZTaxDebitNEx(EZTaxSession session, struct NPANXX_codeEx *trans, int
                                      *err_code);
PF_FUNC_SPEC short int EZTaxDebitPEEx(EZTaxSession session, struct P_CodeEx *trans, int
                                       *err_code);
PF_FUNC_SPEC short int EZTaxDebitTaxInclusiveJCode(EZTaxSession session, struct J_CodeEx
                                                   *trans, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxDebitTaxInclusiveNPAN(EZTaxSession session, struct NPANXX_codeEx
                                                   *trans, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxDebitTaxInclusivePCode(EZTaxSession session, struct P_CodeEx
                                                   *trans, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxDebitTaxInclusiveZip(EZTaxSession session, struct zip_codeEx
                                                   *trans, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxDebitZEx(EZTaxSession session, struct zip_codeEx *trans, int
                                      *err_code);

/* Credit / Adjustment Tax Calculation Prototypes*/
PF_FUNC_SPEC short int EZTaxAdjDebitJEx(EZTaxSession session, struct J_CodeEx *trans, int
                                         discount_type, int adj_method, int *err_code);

```

```

PF_FUNC_SPEC short int EZTaxAdjDebitNEx(EZTaxSession session, struct NPANXX_codeEx *trans, int
discount_type, int adj_method, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjDebitPEx(EZTaxSession session, struct P_CodeEx *trans, int
discount_type, int adj_method, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjDebitTaxInclusiveJCode(EZTaxSession session, struct J_CodeEx
*trans, int discount_type, int adj_method, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjDebitTaxInclusiveNPAN(EZTaxSession session, struct
NPANXX_codeEx *trans, int discount_type, int adj_method, double *base_sale, int
*err_code);
PF_FUNC_SPEC short int EZTaxAdjDebitTaxInclusivePCode(EZTaxSession session, struct P_CodeEx
*trans, int discount_type, int adj_method, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjDebitTaxInclusiveZip(EZTaxSession session, struct zip_codeEx
*trans, int discount_type, int adj_method, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjDebitZEx(EZTaxSession session, struct zip_codeEx *trans, int
discount_type, int adj_method, int *err_code);

/* API Override Prototypes */
PF_FUNC_SPEC short int EZTaxOvrJCodeEx(EZTaxSession session, unsigned long int j_code, struct
enhancedOverride *over);
PF_FUNC_SPEC short int EZTaxOvrJCodeWithAdd(EZTaxSession session, unsigned long int j_code,
struct enhancedOverride *over);
PF_FUNC_SPEC short int EZTaxOvrNPANEx(EZTaxSession session, unsigned long int npanxx, struct
enhancedOverride *over);
PF_FUNC_SPEC short int EZTaxOvrPCodeEx(EZTaxSession session, unsigned long int p_code, struct
enhancedOverride *over);
PF_FUNC_SPEC short int EZTaxOvrZipEx(EZTaxSession session, struct zip_address_p4 zip_addr,
struct enhancedOverride *over, int *err_code);
PF_FUNC_SPEC short int EZTaxLogicOvrJCodeEx(EZTaxSession session, unsigned long int j_code,
struct logicOverride *over);
PF_FUNC_SPEC short int EZTaxLogicOvrJCodeExP(EZTaxSession session, unsigned long int j_code,
struct logicOverrideP *over);
PF_FUNC_SPEC short int EZTaxOldOvrJCodeEx(EZTaxSession session, unsigned long int j_code,
struct tax_ovrd *over);
PF_FUNC_SPEC short int EZTaxSetSafeHarborTAMOverride(EZTaxSession session,
short int safe_harbor_type, double original_federal_tam, double new_federal_tam, int
*err_code);
PF_FUNC_SPEC short int EZTaxClearSafeHarborTAMOverride(EZTaxSession session,
short int safe_harbor_type, int *err_code);

/* Standard Tax Calculation Prototypes*/
PF_FUNC_SPEC short int EZTaxJCodeEx(EZTaxSession session, struct J_CodeEx *trans, int
*err_code);
PF_FUNC_SPEC short int EZTaxNPANEx(EZTaxSession session, struct NPANXX_codeEx *trans, int
*err_code);
PF_FUNC_SPEC short int EZTaxPCodeEx(EZTaxSession session, struct P_CodeEx *trans, int
*err_code);
PF_FUNC_SPEC short int EZTaxProRateJCodeEx(EZTaxSession session, struct J_CodeEx *trans,
double percent, int *err_code);
PF_FUNC_SPEC short int EZTaxProRateNPANEx(EZTaxSession session, struct NPANXX_codeEx *trans,
double percent, int *err_code);
PF_FUNC_SPEC short int EZTaxProRatePCodeEx(EZTaxSession session, struct P_CodeEx *trans,
double percent, int *err_code);
PF_FUNC_SPEC short int EZTaxProRateThisJCodeEx(EZTaxSession session, struct this_J_CodeEx
*trans, double percent, int *err_code);
PF_FUNC_SPEC short int EZTaxProRateThisPCodeEx(EZTaxSession session, struct this_P_CodeEx
*trans, double percent, int *err_code);
PF_FUNC_SPEC short int EZTaxProRateZipEx(EZTaxSession session, struct zip_codeEx *trans,
double percent, int *err_code);
PF_FUNC_SPEC short int EZTaxTaxInclusiveJCode(EZTaxSession session, struct J_CodeEx *trans,
double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxTaxInclusiveNPAN(EZTaxSession session, struct NPANXX_codeEx
*trans, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxTaxInclusivePCode(EZTaxSession session, struct P_CodeEx *trans,
double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxTaxInclusiveZip(EZTaxSession session, struct zip_codeEx *trans,
double *base_sale, int *err_code);

```

```

PF_FUNC_SPEC short int EZTaxTPPEx(EZTaxSession session, struct TPP_addrEx *tpp_trans, struct
    nexus_table *nex_tab, short int table_length, int *err_code);
PF_FUNC_SPEC short int EZTaxZipEx(EZTaxSession session, struct zip_codeEx *trans, int
    *err_code);
PF_FUNC_SPEC short int EZTaxThisJCodeEx(EZTaxSession session, struct this_J_CodeEx *trans, int
    *err_code);
PF_FUNC_SPEC short int EZTaxThisPCodeEx(EZTaxSession session, struct this_P_CodeEx *trans, int
    *err_code);
PF_FUNC_SPEC short int EZTaxPrivateLine(EZTaxSession session, struct private_line_P_Code
    *trans, int *err_code);
PF_FUNC_SPEC short int EZTaxPrivateLineAdj(EZTaxSession session, struct private_line_P_Code
    *trans, int discount_type, int adj_method, int *err_code);

/* Bridge Conferencing Prototypes */
PF_FUNC_SPEC short int EZTaxBridgeConference(EZTaxSession session, struct
    BridgeConferenceTransaction* bconft, struct BridgeConferenceTaxes** pbctaxes, int
    *err_code);
PF_FUNC_SPEC void FreeBridgeParticipantMemory(struct BridgeConferenceTaxes* pBCT);

/* Credit / Adjustment Tax Calculation Prototypes*/
PF_FUNC_SPEC short int EZTaxAdjTaxInclusiveJCode(EZTaxSession session, struct J_CodeEx *trans,
    int discount_type, int adj_method, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjTaxInclusiveNPAN(EZTaxSession session, struct NPANXX_codeEx
    *trans, int discount_type, int adj_method, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjTaxInclusivePCode(EZTaxSession session, struct P_CodeEx *trans,
    int discount_type, int adj_method, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjTaxInclusiveZip(EZTaxSession session, struct zip_codeEx *trans,
    int discount_type, int adj_method, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjJCodeEx(EZTaxSession session, struct J_CodeEx *trans, int
    discount_type, int adj_method, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjNPANEx(EZTaxSession session, struct NPANXX_codeEx *trans, int
    discount_type, int adj_method, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjPCodeEx(EZTaxSession session, struct P_CodeEx *trans, int
    discount_type, int adj_method, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjProRateJCode(EZTaxSession session, struct J_CodeEx *trans, int
    discount_type, double percent, int adj_method, int prorate_credit_or_cancel,int
    *err_code);
PF_FUNC_SPEC short int EZTaxAdjProRateNPAN(EZTaxSession session, struct NPANXX_codeEx *trans,
    int discount_type, double percent, int adj_method, int prorate_credit_or_cancel,int
    *err_code);
PF_FUNC_SPEC short int EZTaxAdjProRatePCode(EZTaxSession session, struct P_CodeEx *trans, int
    discount_type, double percent, int adj_method, int prorate_credit_or_cancel,int
    *err_code);
PF_FUNC_SPEC short int EZTaxAdjProRateThisJCode(EZTaxSession session, struct this_J_CodeEx
    *trans, int discount_type, double percent, int adj_method, int
    prorate_credit_or_cancel,int *err_code);
PF_FUNC_SPEC short int EZTaxAdjProRateThisPCode(EZTaxSession session, struct this_P_CodeEx
    *trans, int discount_type, double percent, int adj_method, int
    prorate_credit_or_cancel,int *err_code);
PF_FUNC_SPEC short int EZTaxAdjProRateZip(EZTaxSession session, struct zip_codeEx *trans, int
    discount_type, double percent, int adj_method, int prorate_credit_or_cancel,int
    *err_code);
PF_FUNC_SPEC short int EZTaxAdjTPPEx(EZTaxSession session, struct TPP_addrEx *tpp_trans, int
    discount_type, struct nexus_table *nex_tab, short _count, int adj_method, int
    *err_code);
PF_FUNC_SPEC short int EZTaxAdjZipEx(EZTaxSession session, struct zip_codeEx *trans, int
    discount_type, int adj_method, int *err_code);

/* Tax Log Prototypes */
PF_FUNC_SPEC struct EZTax_logEx *EZTaxGetCustomLog(EZTaxSession session);
PF_FUNC_SPEC short int EZTaxGetCustomLogCount(EZTaxSession session);
PF_FUNC_SPEC short int EZTaxWriteToLogEx(EZTaxSession session, int count, struct EZTax_logEx
    *log);
PF_FUNC_SPEC struct EZTax_log_v914 *EZTaxGetLogV914(EZTaxSession session);
PF_FUNC_SPEC short int EZTaxGetLogV914Count(EZTaxSession session);
PF_FUNC_SPEC short int EZTaxWriteToLogV914(EZTaxSession session, int count, struct
    EZTax_log_v914 *log);
/* The flush function works for all log types */
PF_FUNC_SPEC short int EZTaxFlushToLogEx(EZTaxSession session

```

```

/* Deprecated Prototypes (preserved for backwards compatibility) */
PF_FUNC_SPEC short int EZTaxAdjProRateJCodeEx(EZTaxSession session, struct J_CodeEx *trans,
                                              int discount_type, double percent, int adj_method, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjProRateNPANEx(EZTaxSession session, struct NPANXX_codeEx
                                              *trans, int discount_type, double percent, int adj_method, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjProRatePCodeEx(EZTaxSession session, struct P_CodeEx *trans,
                                              int discount_type, double percent, int adj_method, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjProRateThisJCodeEx(EZTaxSession session, struct this_J_CodeEx
                                              *trans, int discount_type, double percent, int adj_method, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjProRateThisPCodeEx(EZTaxSession session, struct this_P_CodeEx
                                              *trans, int discount_type, double percent, int adj_method, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjProRateZipEx(EZTaxSession session, struct zip_codeEx *trans,
                                             int discount_type, double percent, int adj_method, int *err_code);
PF_FUNC_SPEC short int EZTaxDebitRevJCode(EZTaxSession session, struct J_CodeEx *trans, double
                                         *base_sale, int *err_code); //Replaced by EZTaxDebitTaxInclusiveJCode
PF_FUNC_SPEC short int EZTaxDebitRevNPAN(EZTaxSession session, struct NPANXX_codeEx *trans,
                                         double *base_sale, int *err_code); //Replaced by EZTaxDebitTaxInclusiveNPAN
PF_FUNC_SPEC short int EZTaxDebitRevPCode(EZTaxSession session, struct P_CodeEx *trans, double
                                         *base_sale, int *err_code); //Replaced by EZTaxDebitTaxInclusivePCode
PF_FUNC_SPEC short int EZTaxDebitRevZip(EZTaxSession session, struct zip_codeEx *trans, double
                                         *base_sale, int *err_code); //Replaced by EZTaxDebitTaxInclusiveZip
PF_FUNC_SPEC short int EZTaxAdjDebitRevJCode(EZTaxSession session, struct J_CodeEx *trans, int
                                             discount_type, int adj_method, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjDebitRevNPAN(EZTaxSession session, struct NPANXX_codeEx *trans,
                                             int discount_type, int adj_method, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjDebitRevPCode(EZTaxSession session, struct P_CodeEx *trans, int
                                             discount_type, int adj_method, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjDebitRevZip(EZTaxSession session, struct zip_codeEx *trans, int
                                             discount_type, int adj_method, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxRevJCode(EZTaxSession session, struct J_CodeEx *trans, double
                                         *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxRevNPAN(EZTaxSession session, struct NPANXX_codeEx *trans, double
                                         *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxRevPCode(EZTaxSession session, struct P_CodeEx *trans, double
                                         *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxRevZip(EZTaxSession session, struct zip_codeEx *trans, double
                                         *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjRevJCode(EZTaxSession session, struct J_CodeEx *trans, int
                                         discount_type, int adj_method, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjRevNPAN(EZTaxSession session, struct NPANXX_codeEx *trans, int
                                         discount_type, int adj_method, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjRevPCode(EZTaxSession session, struct P_CodeEx *trans, int
                                         discount_type, int adj_method, double *base_sale, int *err_code);
PF_FUNC_SPEC short int EZTaxAdjRevZip(EZTaxSession session, struct zip_codeEx *trans, int
                                         discount_type, int adj_method, double *base_sale, int *err_code);

/* If defined for C++ environment */
#ifndef __cplusplus
}
#endif

#endif /* _inc_EZTaxProto_ */

```

8.4 Appendix G Monthly Update Procedure

The AFC monthly update is available at approximately 4:00 PM Central time on the day before the last business day of each month. It contains updated tax information and database files and is available at the Avalara Support web site. The update contains changes resulting from ongoing research and development, providing the most current and efficient tax-rating engine available.

The process of updating your AFC system is similar to the procedure used to install AFC. Use the Installation Wizard to update your files monthly. There is no need to compile your system data because the Avalara database and AFC software functions are independent from your billing system. Simply follow the seven easy steps.

WARNING

AFC **MUST** be updated each month in order for the program to be current. If the download is not completed within the allotted month, AFC is disabled to prevent incorrect tax calculations.

8.4.1 Download the Monthly Update

The Monthly updates are available from the client download page of the **AFC Comms Platform** web site: <https://communications.avalara.net>.

1. Click on this URL or copy it to your browser and click “go.” You will be taken to the Avalara Login Screen. Refer to Figure 8-1.
2. Enter your User ID and Password and click on the “Log In” button.

Figure 8-1 Web Page Login Screen

The screenshot shows the Avalara login page. At the top, there's a logo and the text "AvaTax for Communications Tax Support". Below that, a slogan says "We make it easy.". There are two input fields: one for "Username *" and one for "Password *". To the right of the password field is a link: "Please contact us if you need assistance accessing your account". At the bottom left is a blue "Log in" button.

- Refer to Figure 8-2. After a successful Login, click on “Downloads” in the selection menu near the top of the screen.

Figure 8-2 Support Home



Avalara for Communications (AFC a.k.a EZtax) Support Home

Welcome to the Avalara for Communications (AFC a.k.a EZtax) support web site. This site provide information and updates related to AFC products and services.

To download the monthly updates for AFC and AFC Manager, click on the Downloads link in the top navigation bar. If you have any difficulty downloading the files, please contact communicationsupport@avalara.com.

Be sure to download the current Tax Change Notices to get important information on new jurisdictions, the new DLL/shared object and taxing VoIP products and services.

- Refer to Figure 8-3. The Downloads screen contains the list of file(s) (with descriptions and versions) to be downloaded. Click on the description of the file to be downloaded.

Figure 8-3 Downloads Available



Available Downloads

Description	Version	Locked	Download
Update Notices	1605		
EZTax 9.0 (AS/400 Platform)	1605		
EZTax 9.0 (DGUX Platform)	1605		
EZTax 9.0 (HPUX 32-bit Platform)	1605		
EZTax 9.0 (HPUX 64-bit Platform)	1605		
EZTax 9.0 (Red Hat Linux 32-bit Platform)	1605		
EZTax 9.0 (Sun 32-bit Platform)	1605		
EZTax 9.0 (Sun 64-bit Platform)	1605		
EZTax 9.0 (Windows Platform)	1605		
EZTax 9.0 (Windows Platform - Sage)	1605		
EZTax SAU 9.1 (Windows SAU 32-bit Platform)	1605		
EZTax 9.0 (Windows 64-bit Platform)	1605		

5. At the bottom of the screen, a window asking if you want to open or save the selected file appears. Click “Open” and the content of the file is opened. Save the file in the appropriate location.

6. Refer to Figure 8-4. After the file has been downloaded you will be returned to the Download screen. There will be a check mark next to the file you downloaded. Mouse-over the check mark to view the User Name and time that the file was downloaded.

Figure 8-4 Downloaded Files

The screenshot shows the Avalara AvaTax for Communications Tax Support download page. The top navigation bar includes links for Home, Downloads, Notices, and a welcome message for 'QA testing account'. The main content area is titled 'Available Downloads' and displays a table of software packages:

Description	Version	Locked	Download
Update Notices	1605		✓ ✓ ✓ ✓
EZTax 9.0 (AS/400 Platform)	1605		✓ ✓ ✓
EZTax 9.0 (DGUX Platform)	1605		
EZTax 9.0 (HPUX 32-bit Platform)	1605		
EZTax 9.0 (HPUX 64-bit Platform)	1605		
EZTax 9.0 (Red Hat Linux 32-bit Platform)	1605		✓ ✓
EZTax 9.0 (Sun 32-bit Platform)	1605		✓ ✓
EZTax 9.0 (Sun 64-bit Platform)	1605		✓ ✓
EZTax 9.0 (Windows Platform)	1605		✓ ✓ ✓
EZTax 9.0 (Windows Platform - Sage)	1605		✓ ✓
EZTax SAU 9.1 (Windows SAU 32-bit Platform)	1605		✓ ✓ ✓
EZTax 9.0 (Windows 64-bit Platform)	1605		✓ ✓

7. If you have custom designed your application with different directories than the defaults established by Avalara then move the appropriate files to their folders.

8.4.2 Important Files in Update

The following files and directories are included in the monthly update:

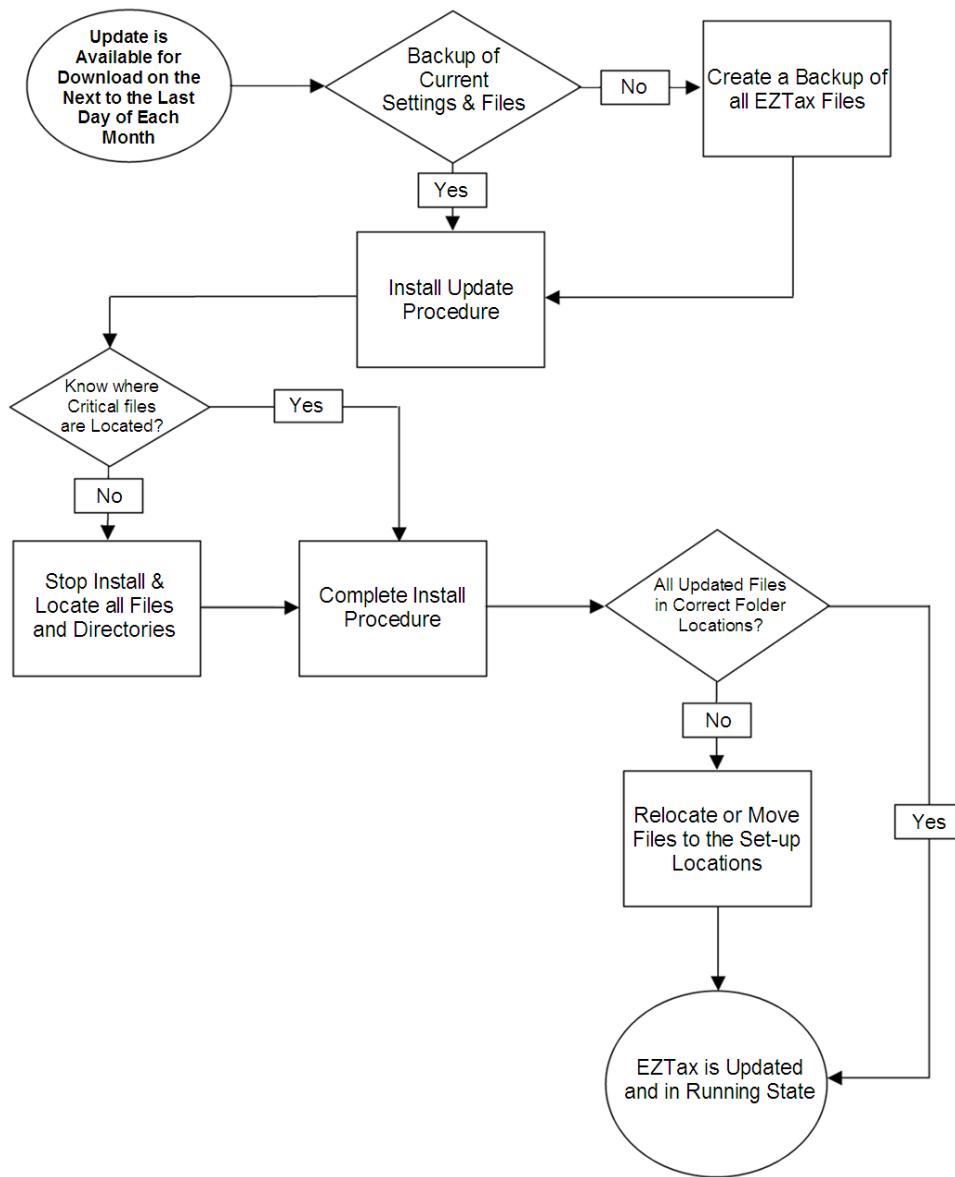
- Update.exe - Utility that steps the user through the process of installing the update.
- Current tax changes and other documents - located in the \support\docs directory are file format keys, guidelines, and the Tax Changes <month year>.doc
- filelocs.upd – rename this new file “filelocs.txt”

8.4.3 Monthly Maintenance

Proper monthly maintenance will keep the AFC system operating without interruption. Refer to the following figure for the update procedure flow.

1. Take an inventory of all file directories and locations.
2. Create a backup of these either on media such as a diskette or save them to a backup directory.
3. Once the update is installed, closely review the update file locations.
4. Run a Test on a small file.
5. Review the \support\docs directory for tax changes and current update information.

Figure 8-5 EZtax Update Procedure



8.5 Help Guide

The Help guide is provided to support users when encountering difficulties with AFC. Users are encouraged to contact Avalara with concerns of any nature:

Toll Fee: 800-525-8175

Corporate Website: <http://communications.avalara.com/>

AFC Comms Platform Website: <https://communications.avalara.net>

Email: communicationsupport@avalara.com

8.5.1 Troubleshooting

Troubleshooting information is provided to assist users in diagnosing and resolving issues when encountered.

Table 8-1 Troubleshooting

Issue	Solution
When running an application that uses the EZTax2.dll, get Unable to locate DLL error.	Make sure that EZTax2.dll is located in the working directory or in a directory that is in your search path.
Open of filelocs.txt configuration file failed!	Place the filelocs.txt file in your working directory
EZTaxInitEx function call returns null tax table pointer and invalid session handle.	There is number of reasons EZTaxInitEx failed. Look in the *.sta status files for indication of why the failure happened. Most common items are the Database files are expired or the file paths to the database and output files were not specified.
When making adjustments, AFC returns a positive tax.	When using the AFC adjustment functions, make sure the tax amount entered is positive. The adjustment functions will then negate the charge or lines.
I calculate taxes by hand using the charge I sent and the tax rate and tax amount AFC returns and the results do not compute.	AFC takes into account tax on taxes, which will make the tax base larger than the charge that was passed to AFC.

8.5.2 FAQ's

FAQ's are provided to answer common user questions.

Table 8-2 FAQ's

Question	Answer
What happens if I passed in an invalid NPANXX?	No taxes are generated and the invalid number is reported to EZTax.sta, and the err_code is set.
How do I apply taxes to other countries by using the NPANXX functions?	By passing into the system the country id for that country as the NPANXX. This should be defined in a header file.
I want to call AFC to compute taxes for a quote but not log the taxes to the file.	Open a second session with logging turned off.
What if I want to get a PCode for a Country or State in AFC?	Use the EZTaxZipToPCodeP4Ex function. If trying to find PCode for a country, specify the country ISO code and the rest of the field as null strings. If trying to find PCode for a state, specify the country ISO code, state abbreviation and the rest of the fields as null strings.
Do I need to input the county for EZTaxZip functions?	No, but if city crosses county boundaries, AFC will return the first match it finds.
I have more than one tax log for the month. How do I produce one compliance report?	You can use EZTaxAppend or EZTaxAppendF utilities to append binary tax logs together. Also some AFC reporting utilities allow combining tax logs together.
Do the field lengths include the null terminator for C strings.	All field lengths are the actual size in the structure. Be sure to allow one character for the null terminator.