



# **AvaTax for Communications**

## **SaaS Pro Sales and Use Developer Manual**

Release: 9.19.1802.1  
Document: TM\_00117\_0043  
Date: 02/15/2018

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

## 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
02/19/2016	0026	Avalara branding updates to reflect the transition to the new company and product names have been incorporated into this document. Please see <a href="#">Appendix A – Avalara Product Names</a> for specific changes in product references and descriptions.
05/18/2016	0027	Updates to support Custom Log and Category Exemptions include: Addition of new <b>Section 1.9 Exemptions</b> , added new <b>Section 4 Generating Custom Reports</b> , updates to <b>Section 8 Utility Web Methods</b> (CreateReport) and updates to <b>Section 9 Web Service Data Definitions</b> (ReportOptions, CustomLogField and CategoryExemption tables)
06/17/2016	0028	Added new <b>Section 1.13 Tax Grouping</b> , <b>Section 3.6 Supported APIs</b> for Customer Mode, updated <b>Section 2</b> with details for entry of Canadian postal codes, added note to <b>Section 6.18 SAUCalcJurisdiction</b> , updated <b>Section 7 Sales and Use Customer Mode Web Methods</b> with notice of 3 SAU APIs to be deprecated in the near future.
08/02/2016	0029	Added note in <b>Section 1.8 Exclusions</b> regarding excluded tax jurisdictions appearing as unknown or not appearing on TSRs. Added new <b>Section 5 Optional Fields</b> to describe default and extended optional fields. Updated <b>Section 9 Utility Web Methods</b> with new APIs, <b>GetOptionalFieldKeyDesc</b> and <b>UpdateOptionalFieldKeyDesc</b> to support additional alphanumeric field for AFC SaaS Pro. OptionalField added to Transaction table and an OptionalField and OptionalKey table added in <b>Section 10 Web Service Data Definitions</b> .
09/26/2016	0030	Updated <b>Section 4.4 Custom Log Report Columns</b> to include columns for the Extended Optional fields. Also, added new <b>Section 12 Monthly Update Procedures</b> for downloading documentation.
11/15/2016	0031	Updated description in <b>Section 1.9.2 Level Exemptions</b> . Updated single transaction maximum from 3,000 to 50,000 in <b>Section 3 Customer Mode</b> . Added new <b>Section 3.7 Tax Inclusive Transactions</b> to indicate how to process tax inclusive calculations in customer mode. Added new <b>Section 6 Zip Lookup Requests</b> and <b>10.16 Zip Lookup</b> to provide an overview of zip lookup functionality and use of the API. Added new <b>Section 8.17 ProcessCustomerBatchV2</b> API and <b>Section 8.19 CalcTaxesInCustModeV2</b> API. Updated <b>Section 11 Web Data Definitions</b> to include updated data structure tables for <b>SalesUseTransaction</b> , and new ones for <b>TaxDataV2</b> , <b>CustomerResultsV2</b> , <b>ZipLookup</b> , <b>ZipLookupResults</b> and <b>LocationData</b> .
12/05/2016	0032	Updated <b>Section 3 Customer Mode</b> with note providing recommendation on appropriate duration of a transaction in invoice mode. Updated <b>Section 4.6.1 Accessing the FTP Site</b> to reflect utilization of a secure file-transfer protocol (SFTP) in addition to FTP. Updated remarks in <b>Section 10.7 ZipToPCODE</b> .
01/11/2017	0033	Release, version numbers and copyright year updated.
01/18/2017	0034	Updated <b>Section 3.6 Supported and Non-Supported Features</b> (under Customer Mode).
02/13/2017	0035	Added new <b>Section 1.1 Getting Started</b> which includes requirements for integration and key inputs. Updated <b>Section 1.4 Transaction Date</b> with additional details for date format.

Date	Version	Description
03/13/2017	0036	All references to 'reverse' tax calculations in AFC SaaS Pro have been updated and renamed to reflect the current naming convention which is 'tax inclusive' calculations. As a result, 6 APIs in <b>Section 8 Sales and Use Web Methods</b> have been updated and renamed. Updated <b>Section 1.10.3 Specific Exemptions</b> with note regarding availability of wildcard character for tax types. Release number and version updates. Added new <b>Section 1.15 Specifying a Unique Identifier</b> . Updated description in <b>Discount Table</b> in <b>Section 12 Web Service Enumeration Definitions</b> .
04/14/2017	0037	Added <b>Section 1.16 Commit/Uncommit</b> . and <b>Section 1.17 Application of Tax Brackets and Limits</b> . All references to 'Customer Mode' in AFC SaaS Pro have been updated and renamed to reflect the current naming convention which is 'Invoice Mode.' As a result, <b>Section 3</b> has been updated and renamed <b>Invoice Mode</b> .
05/25/2017	0038	Updated <b>Section 1.3 Client Information</b> . Added <b>Section 8.29</b> for <b>CommitTransactions</b> API. Added <b>CommitTransactions</b> to the table in <b>Section 8 SOAP Telecom Web Methods</b> . Added <b>CommitData</b> data structure table to <b>Section 11 Web Service Data Definitions</b> .
06/27/2017	0039	Updated links in <b>Section 1.1 Getting Started</b> and updated <b>Section 13 Monthly Update</b> .
07/07/2017	0040	Miscellaneous updates in <b>Section 3 Invoice Mode</b> .
10/25/2017	0041	Updated Avalara contact information (address and support site). Removed <b>Appendix A – Avalara Product Names</b> .
11/22/2017	0042	Added note related to timestamp/invoice date passed in <b>Section 1.4 Transaction Date</b> . Updated <b>Section 2 Specifying a Tax Jurisdiction</b> with a note and table providing overview of alternate formats for US zip code plus 4 digits. Updated <b>Section 12 Web Service Enumeration Definitions</b> with an <b>ExemptionType</b> table which provides values and descriptions.
02/15/2018	0043	Updated definition for surcharge in <b>Section 4.4 Custom Log Report Columns</b> as well as in various tables throughout <b>Section 11 Web Service Data Definitions</b> .

## Table of Contents

<b>1.</b>	<b>Introduction.....</b>	<b>1</b>
1.1	Getting Started .....	2
1.1.1	Integration Requirements .....	2
1.1.2	Key Input Requirements .....	2
1.2	AFC Transaction and Service Types .....	3
1.3	Client Information .....	3
1.4	Transaction Date .....	3
1.5	Tax Types.....	4
1.6	Bundles .....	5
1.7	Overrides.....	5
1.8	Nexus .....	5
1.9	Exclusions.....	6
1.10	<b>Exemptions.....</b>	<b>6</b>
1.10.1	Category Exemptions.....	6
1.10.2	Level Exemptions.....	7
1.10.3	Tax Type Exemptions.....	7
1.11	Compliance Files .....	7
1.12	Live vs. Test Transactions .....	9
1.13	Compliance Month.....	10
1.14	Tax Grouping.....	10
1.15	Specifying a Unique Identifier .....	12
1.16	Commit/Uncommit .....	13
1.17	<b>Application of Tax Brackets and Limits.....</b>	<b>13</b>
1.17.1	Tax Brackets .....	13
1.17.2	Tax Limits.....	13
<b>2.</b>	<b>Specifying a Tax Jurisdiction.....</b>	<b>14</b>
2.1	Getting the Right Tax Jurisdiction for Local Taxation .....	15
<b>3.</b>	<b>Invoice Mode .....</b>	<b>16</b>
3.1	Overview .....	16
3.2	Batch Transaction Submission.....	17

3.3	Single Transaction Submission .....	17
3.4	Taxing Jurisdiction Specification.....	17
3.5	API Results.....	18
3.6	Supported and Non-Supported Features .....	18
3.7	<b>Tax Inclusive Transactions.....</b>	<b>18</b>
3.7.1	Tax Inclusive - Batch Transaction Submission.....	19
3.7.2	Tax Inclusive - Single Transaction Submission .....	19
4.	<b>Generating Custom Reports.....</b>	<b>20</b>
4.1	Report Process Overview .....	20
4.2	Setting up ReportOptions.....	20
4.2.1	BaseReport .....	20
4.2.2	CreateNbaFile .....	20
4.2.3	CreateNcaFile .....	21
4.2.4	CustomLogFields .....	21
4.2.5	EmailAddress .....	21
4.2.6	FileExtension .....	22
4.2.7	FileName .....	22
4.2.8	IncludeHeaders.....	22
4.2.9	Precision .....	23
4.2.10	ReportType.....	23
4.2.11	StartDate, EndDate, and TimeSpan .....	23
4.3	Report Types .....	23
4.4	Custom Log Report Columns.....	24
4.5	Aggregation.....	26
4.6	Output Files.....	27
4.6.1	Accessing the FTP Site .....	27
4.7	Sample Code .....	29
5.	<b>Optional Fields .....</b>	<b>30</b>
5.1	Overview .....	30
5.2	Default Optional Fields.....	30
5.3	Extended Optional Fields .....	30

5.3.1	Setting Up an Extended Optional Field in a Transaction.....	31
5.3.2	UpdateOptionalFieldKeyDesc.....	31
5.3.3	GetOptionalFieldKeyDesc.....	32
6.	<b>Zip Lookup Requests .....</b>	<b>32</b>
6.1	<b>Overview .....</b>	<b>32</b>
6.2	<b>Setting up a Request.....</b>	<b>32</b>
6.2.1	Best Match .....	32
6.2.2	Location Data .....	33
6.2.3	Limit Results .....	33
6.3	<b>Results.....</b>	<b>33</b>
6.4	<b>Examples .....</b>	<b>33</b>
7.	<b>AFC SaaS Pro Sales and Use Web Service Programmer Reference .....</b>	<b>35</b>
7.1	TaxService Endpoint.....	35
8.	<b>Sales and Use Web Methods.....</b>	<b>36</b>
8.1	SAUCalcTaxesWithPCode .....	37
8.2	SAUCalcTaxesWithFipsCode.....	38
8.3	SAUCalcTaxesWithZipAddress.....	39
8.4	SAUCalcAdjWithPCode.....	39
8.5	SAUCalcAdjWithFipsCode .....	40
8.6	SAUCalcAdjWithZipAddress .....	41
8.7	SAUCalcReverseTaxesWithPCode .....	42
8.8	SAUCalcReverseTaxesWithFipsCode .....	42
8.9	SAUCalcReverseTaxesWithZipAddress.....	43
8.10	SAUCalcReverseAdjWithPCode .....	44
8.11	SAUCalcReverseAdjWithFipsCode.....	45
8.12	SAUCalcReverseAdjWithZipAddress.....	46
8.13	SAUCalcTaxInclusiveTaxesWithPCode.....	47
8.14	SAUCalcTaxInclusiveTaxesWithFipsCode .....	47
8.15	SAUCalcTaxInclusiveTaxesWithZipAddress .....	48
8.16	SAUCalcTaxInclusiveAdjWithPCode .....	49
8.17	SAUCalcTaxInclusiveAdjWithFipsCode .....	50

8.18	SAUCalcTaxInclusiveAdjWithZipAddress .....	51
8.19	BeginCustomerBatch.....	51
8.20	SAUCalcCustTaxes.....	52
8.21	SAUCalcCustAdj.....	52
8.22	ProcessCustomerBatch .....	53
8.23	ProcessCustomerBatchV2 .....	54
8.24	CalcTaxesInCustMode .....	54
8.25	CalcTaxesInCustModeV2.....	55
8.26	SAUCalcJurisdiction .....	56
8.27	SAUCalcTaxesWithOverrides.....	56
8.28	SAUCalcAdjWithOverrides .....	57
8.29	CommitTransactions .....	58
8.30	Error Messages Common to all Calculation Methods .....	59
<b>9.</b>	<b>Sales and Use Invoice Mode Web Methods .....</b>	<b>60</b>
9.1	SAUCalcTaxesAndAdjWithPCodeInCustMode.....	60
9.2	SAUCalcTaxesAndAdjWithFipsCodeInCustMode .....	61
9.3	SAUCalcTaxesAndAdjWithZipAddressInCustMode .....	62
<b>10.</b>	<b>Utility Web Methods.....</b>	<b>63</b>
10.1	GetAddress.....	63
10.2	GetTaxCategory.....	64
10.3	GetTaxDescription.....	64
10.4	GetTaxRates .....	65
10.5	FipsToPCode.....	65
10.6	PCodeToFips.....	66
10.7	ZipToPCode .....	66
10.8	NpaNxxToPCode.....	67
10.9	GetServerTime .....	68
10.10	GetVersion .....	68
10.11	GetEZtaxVersion.....	69
10.12	GetEZtaxDbVersion .....	69
10.13	CreateReport.....	69



10.14	GetOptionalFieldKeyDesc.....	70
10.15	UpdateOptionalFieldKeyDesc.....	71
10.16	ZipLookup.....	71
11.	<b>Web Service Data Definitions.....</b>	<b>72</b>
	SalesUseTransaction .....	72
	CountyPCode.....	72
	ZipAddress .....	73
	AddressData.....	74
	Nexus.....	74
	Exclusion .....	74
	TaxExemption .....	74
	TaxData.....	75
	TaxDataV2.....	76
	CustomerResults .....	77
	CustomerResultsV2.....	77
	CustomerTaxData .....	77
	ReverseTaxResults .....	78
	TaxInclusiveTaxResults .....	78
	DefaultAttributes .....	78
	DemurrageAttributes .....	78
	TradeInAttributes.....	78
	DepositAttributes.....	78
	FreightAttributes.....	78
	DiscountAttributes.....	79
	FinanceAttributes.....	79
	InstallationAttributes .....	79
	ShipAndHandlingAttributes.....	79
	SoftwareMaintAttributes .....	79
	ServiceContractAttributes.....	79
	MaintAgreementAttributes.....	79
	FactoryWarrantyAttributes .....	80

ExtWarrantyAttributes .....	80
TaxRateInfo .....	80
TaxRateHistory .....	80
TaxBracketInfo .....	80
TaxRateOverrideInfo .....	80
ReportOptions .....	81
CustomLogField .....	81
CategoryExemption .....	82
OptionalField .....	82
OptionalKey .....	82
ZipLookup .....	82
ZipLookupResult .....	83
LocationData .....	83
CommitData .....	83
<b>12. Web Service Enumeration Definitions .....</b>	<b>84</b>
12.1 Specifying Tax Level .....	84
TaxLevel .....	84
12.2 Specifying Calculation Type .....	84
CalculationType .....	84
12.3 Specifying Sale or Consumed .....	84
SaleOrConsumed .....	84
12.4 Specifying FOB .....	85
FOB .....	85
12.5 Specifying Adjustment Method .....	85
AdjustmentMethod .....	85
12.6 Specifying Discount Type .....	85
SalesDiscountType .....	85
12.7 Specifying Finance Type .....	86
FinanceType .....	86
12.8 Specifying Installation Type .....	87
InstallationType .....	87

12.9	Specifying Software Agreement Type .....	87
	<b>SoftwareAgreementType</b> .....	87
12.10	Specifying Service Contract Agreement Type .....	88
	<b>ServiceContractAgreementType</b> .....	88
12.11	Specifying Maintenance Agreement Type .....	88
	<b>MaintenanceAgreementType</b> .....	88
12.12	Specifying Factory Warranty Type .....	89
	<b>FactoryWarrantyType</b> .....	89
12.13	Specifying Extended Warranty Type .....	89
	<b>ExtendedWarrantyType</b> .....	89
12.14	Specifying Update Type.....	89
	<b>UpdateType</b> .....	89
12.15	Specifying Customer Type.....	90
	<b>CustomerType</b> .....	90
12.16	Specifying Time of Sale.....	90
	<b>TimeOfSale</b> .....	90
12.17	Specifying Item Type .....	90
	<b>ItemType</b> .....	90
12.18	Specifying Exemption Type .....	90
13.	Monthly Update.....	91

## 1. Introduction

AvaTax for Communications (AFC) SaaS Pro is easily integrated into your application. Typically, integration efforts are measured in days, instead of the months required of other products. Avalara provides a complete sales and use taxation solution with its AFC SaaS Pro product. AFC SaaS Pro performs transaction processing for numerous types of products and services, from alcohol to prescription drugs, from general merchandise to software sales.

AFC SaaS Pro relieves or minimizes your organization of the following cost of doing business burdens:

- 1) Cost of research and maintenance of the continually changing tax data of approximately 70,000 communities capable of levying taxes in the United States as well as international taxes.
- 2) Tax compliance reduces liability and exposure to tax audits.
- 3) Cost of tracking, maintenance, and implementation of tax systems based upon the whim of federal, state, county, and/or local governments.
- 4) Cost of research and maintenance of nearly 400,000 tax rates and complex application rules including, but not limited to the following:
  - Over 200 tax types.
  - Federal, state, county, district and local taxes.
  - Taxing for eighty other countries.
  - Applicability to sale and/or resale
  - Applicability to different customer types
  - Taxes that are included in base taxable amount of other taxes
  - Maximum tax bases
  - Excess tax rates
  - Transit Taxes
  - State and county taxes that are replaced by county and /or local taxes
  - Special state and county rates based upon county or local jurisdiction
  - Determination of 1 of 10 different jurisdiction determination rules per tax
  - Maintenance of address to tax jurisdiction cross reference data
  - Application of taxes based upon transaction type
  - Application of taxes based upon service type
  - Application of taxes based upon attribute type

## 1.1 Getting Started

### 1.1.1 Integration Requirements

Please reference the links provided below for the most current and detailed integration requirements.

- Integration Checklist:  
<https://developer.avalara.com/certification/communications/>
- Certification Use Cases:  
<https://developer.avalara.com/certification/communications-certification/>

### 1.1.2 Key Input Requirements

The following items are required as input for AFC SaaS Pro:

- Transaction Date (Please reference [SalesUseTransaction](#) for additional details).
- Jurisdiction information details: (Please reference [Specifying a Tax Jurisdiction](#) for additional details).
  - Address
  - Zip
  - Zip+4
  - FIPS Code
  - PCode
  - NPANXX
- Product Information/ Transaction and Service Types (Please reference [AFC Transaction and Service Types](#) for additional details.)
- Sale Type details: (Please reference [SalesUseTransaction](#) for additional details).
  - Retail
  - Wholesale
- Customer Type Info (of who is being billed) details: (Please reference [CustomerType](#) for additional details).
  - Business
  - Residential
  - Industrial
  - Senior Citizen
- Business Class (of who is billing) details: (Please reference [SalesUseTransaction](#) for additional details).
  - CLEC
  - ILEC

- Other
- Service Class (of who is billing) details: (Please reference [SalesUseTransaction](#) for additional details.)
- - Primarily Long Distance
  - Primarily Local
  - Other
- Other Company Info (of who is billing) details: (Please reference [SalesUseTransaction](#) for additional details.)
  - Regulated: Y or N
  - Franchise: Y or N
  - Facilities: Y or N

\*Certain Inputs can be set as defaults and do not have to be selectable options in the billing platform being utilized. This represents a selection of data elements. Additional features/functions may require additional elements.

## 1.2 AFC Transaction and Service Types

The AFC SaaS Pro software package provides a wide range of transaction and service types. The result is a complete taxation package. When passing a transaction to AFC SaaS Pro, the user must indicate the transaction and service type of the transaction. See **TM\_00506\_AFC Sales and Use Mapping Guidelines.pdf** for more details about the transaction and service types supported by AFC SaaS Pro.

## 1.3 Client Information

Many taxing jurisdictions apply taxes differently to different types of customers. In addition, many jurisdictions apply taxes differently depending upon the type of sale. For these reasons, it is necessary to indicate to AFC SaaS Pro the type of client the transaction occurred with. The client type will be “Business”, “Residential”, “Industrial” or “Senior Citizen”. Likewise, it is necessary to indicate the type of sale, “sale” or “resale”. Transactions that are performed with the end user of that product or service are “sale”. Products that are with a reseller (entity that intends to resell the product or service purchased) are “resale”. To have exempt taxes available for reporting, exemption type 3 (Sales For Resale) should be used in combination with Resale.

## 1.4 Transaction Date

AFC SaaS Pro maintains multiple tax rates for every tax contained in the system. One rate is the current tax rate and the others are the previous tax rates. When transactions are passed to AFC SaaS Pro for tax generation, the caller is required to specify a date. AFC SaaS Pro compares this date to the effective date

of each tax that applies to the transaction. If the date passed to AFC SaaS Pro is "equal to" or "greater than" the effective tax date, the current tax rate is used. If the date passed to AFC SaaS Pro is prior to the effective date, AFC SaaS Pro will select the previous tax rate for the tax based on the effective date of the previous tax rate. If a transaction is passed to AFC SaaS Pro without a date (that is, the date is set to zero), AFC SaaS Pro will set the date to the current date.

The Date field is normally populated with the bill date, invoice date or call date (as applicable) and may appear in one of the following Standard Date and Time Format Strings for .NET Framework shown in the table below.

Date Format	Sample
mm/dd/yyyy	06/01/2016
m/d/yyyy	6/1/2016
mm-dd-yyyy	06-01-2016
m-d-yyyy	6-1-2016
yyyy-mm-dd	2016-06-01
yyyy-m-d	2016-6-1
yyyy-mm-ddThh:MM:ss	2016-06-01T13:45:30
yyyy-m-dThh:MM:ss	2016-6-1T13:45:30

A common question asked for some transactions is "What date should be used for taxing a transaction, the date it occurred or the billing date?" Normally this should be the billing date. Generally accepted accounting principles tell us we should record our liabilities when we record our revenues. In most cases neither of these are recorded (or even known) until billing occurs. Large organizations may have a different answer to this question. If you are running high volumes of transactions daily and record revenue as it occurs, then the tax should be recorded on the same basis (i.e. the transaction date should be used).

**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.

## 1.5 Tax Types

The application of taxes varies from location to location as well as the particular transaction and service provided. Different localities typically have different taxes and logic associated with the application of the taxes. For example, one locality may apply tax to medical equipment paid for by Medicare but not to equipment paid for by Medicaid. A neighboring locality may apply tax to both or neither. For the most current list of Tax Types, see the **TM\_00506\_AFC Sales and Use Mapping Guidelines.pdf** file on the most current Distribution/Update download. This document is under /support/docs.

## 1.6 Bundles

The Bundling feature allows the user to define multiple AFC SaaS Pro transaction/service type pairs as one unique “bundle” transaction/service pair type. AFC SaaS Pro processes all of the transaction/service type pairs in the bundle and stores the results in the file EZtax.bdl, the formatted text bundle definition file. This file must be created and edited by AFC Bundler, an optional utility program. The bundle (.bdl) file must be installed on our web server to be implemented. Email your zipped bundle file to [communicationsupport@avalara.com](mailto:communicationsupport@avalara.com) for installation. We will contact you when it has been installed.

## 1.7 Overrides

Overrides allow the client to change the rate of a tax in the AFC Engine. Avalara markets the AFC Override Utility (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.

### 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.

The override (.ovr) file must be installed on our web server to be implemented. Email your zipped override file to [communicationsupport@avalara.com](mailto:communicationsupport@avalara.com) for installation. We will contact you when it has been installed.

Overrides on tax rates may also be performed for specific transactions.

## 1.8 Nexus

The AFC SaaS Pro Sales and Use interface allows the client to specify the states where the client's company has nexus. There are two ways that nexus can be specified to AFC SaaS Pro:

- (1) Users can provide a nexus file to be installed on our web service. The format of the nexus file is simply a list of state abbreviations, one per line. States that are not listed are assumed to NOT have nexus. The nexus file should be emailed to [communicationsupport@avalara.com](mailto:communicationsupport@avalara.com) for installation. We will contact you when it has been installed.
- (2) Users can provide an array of Nexus objects with each tax method call to the Sales & Use interface. These nexus objects modify the nexus settings only for the current method call.



## 1.9 Exclusions

The AFC Sales & Use interface allows the client to specify the states where the client's company wants to exclude all taxes. There are two ways that exclusions can be specified to AFC SaaS Pro:

- (1) Users can provide an exclusion file to be installed on our web service. The format of the exclusion file is either a state abbreviation or the country ISO code followed by a comma and the state abbreviation, one per line. Example: "AZ" or "USA,AZ" are both valid for excluding taxes in the state of Arizona. The exclusion file should be emailed to [communicationsupport@avalara.com](mailto:communicationsupport@avalara.com) for installation. We will contact you when it has been installed.
- (2) Users can provide an array of Exclusion objects with each tax method call to the Sales & Use interface. These exclusion objects modify the exclusion settings only for the current method call.

If both methods are utilized, the exclusions passed in via the API call will be used and the exclusion file will be ignored for that particular transaction.

**Note:** Excluded tax jurisdictions will either appear as unknown or will not be included in any Transaction Service Reports (TSRs) produced.

## 1.10 Exemptions

There are three types of exemptions allowed in AFC SaaS Pro.

### 1.10.1 Category Exemptions

Category Exemptions are exemptions applied by tax category. (Please reference the table below to view the current listing of tax categories. Users may also reference **Section 6 Category Definitions** in the **AFC Telecom Mapping Guidelines** document for a detailed overview of tax type assignment per tax category.)

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
12	Reserved
13	Value Added Taxes

The Category ID number must be passed in order for exemptions to be applied appropriately. In addition to this, the Country Code or State Abbreviation or both values for the category exemption must be provided as well. Please reference the example provided using the following scenario.

Category	Country	State	Comments
1 – 13	USA		Provided categories will be exempted in all US jurisdictions.
	USA		Error message will be generated as Category is required field.
1 – 13	USA	KS	Provided categories will be exempted in all Kansas jurisdictions.
1 – 13		KS	Provided categories will be exempted in all Kansas jurisdictions by assuming “USA” as country.

### 1.10.2 Level Exemptions

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 level taxes.

### 1.10.3 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 PCode should be specified as tax type 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.

**Note:** To exempt a county or city for all taxes, apply a specified exemption for the county and/or city by entering the tax type as 0. This is the wildcard character for tax type when applying exemptions in AFC SaaS Pro.

## 1.11 Compliance Files

The web service automatically generate compliance files at the end of the month for the clients contracted for compliance services. These files contain a summary of the tax data generated for all transactions processed throughout the month. The format of the compliance files may be that of any of the available AFC SaaS Pro sorting and reporting utilities. The compliance files may be placed in an FTP site where they can be downloaded or they may be automatically emailed to a specific email account. AFC SaaS Pro Clients are required to contact [communicationsupport@avalara.com](mailto:communicationsupport@avalara.com) in order to set up these preferences.

## Sorting & Reporting Utilities

Method	Description	Example
Date Method 0	This method gathers all transactions that were sent to our server during the previous month by looking at start time field for each transaction and generate the compliance files from that data on the first of every month or requested date. AFC SaaS Pro uses date method 0 to generate compliance reports, unless specified by client.	<ul style="list-style-type: none"><li>• <b>Reporting Scenario 1:</b> Throughout the month of January 2013, you submit several transactions to the web service for taxation at the time each sale occurs. The system will generate the January 2013 compliance reports on February 1st, 2013 for all the transactions received in January, irrespective of invoice date.</li></ul>
Date Method 1	This method processes data based on invoice date. We gather all transactions that have an invoice date of the previous month and generate the compliance files from that data on the first day of every month.	<ul style="list-style-type: none"><li>• <b>Reporting Scenario 2:</b> Throughout the month of January 2013, you submit several transactions to the web service for taxation with an invoice date of January 2013. On February 1st, 2013, the system will gather all transactions with an invoice date of January 1st through January 31st 2013 and generate the compliance data.</li><li>• <b>Reporting Scenario 3:</b> You send a transaction with an invoice date of January 15th past February 1st when AFC SaaS Pro runs your compliance reports, this transaction will never show up on any report. You will need to submit a request to our network services to rerun your data for January in order to capture January data sent past February. Additional charges apply to rerun, or provide subsequent compliance reports for the month. Alternatively, you may request your compliance reports to be generated on a specific day of the month in order to give you enough time to complete processing tax calculations for your billing cycle. Please contact <a href="mailto:communicationsupport@avalara.com">communicationsupport@avalara.com</a> in order to set up these preferences.</li></ul>

Method	Description	Example
<b>Date Method 2</b>	This method processes data based on an optional10 field. The client will fill in the optional10 field with the year and month (YYYYMM format) that they want the transaction to show up in their compliance files. When AFC SaaS Pro runs the compliance reports, it only includes transactions from the previous month, based upon optional10 field.	<ul style="list-style-type: none"> <li>• <b>Reporting Scenario 4:</b> Throughout the month of January 2013, you submit several transactions to the web service for taxation at the time each sale occurs. Each transaction must include the value 201301 in the Optional10 field. The system will generate the January 2013 compliance reports on February 1st, 2013 and include the taxes for all transactions with a 201301 value in the Optional10 field. It is important to note that if you send transactions with 201301 in the optional10 field, after the compliance report is run on February 1, they will be provided in any compliance reports. In this case, you should record the transaction as 201302 or a future period for them to be reported upon.</li> <li>• <b>Reporting Scenario 5:</b> On February 1st 2013, you process a batch for all January 2013 transactions. Each transaction must have a 201301 value in the Optional10 field. You should request your reports to be generated on the 2nd day of the month beforehand. The system will generate your January 2013 compliance reports on the 2nd day of the month and include the taxes for all transactions with a 201301 value in the Optional10 field.</li> <li>• <b>Reporting Scenario 6:</b> On February 15th, you process an adjustment for a transaction that took place on January 15<sup>th</sup> 2013. The Date field of the transaction must have the date of the original transaction (January 15<sup>th</sup>) in order to get the appropriate tax adjustments back, but the Optional10 field of the transaction must have a value of 201302 since the adjustment was handled during February. The tax adjustment will be included in your February 2013 compliance reports.</li> </ul>

In order to allow enough time to complete tax calculations for your billing cycle, Network Services will assist in establishing the correct date method depending on your transaction method and the appropriate date of the month for your compliance reports.

Please contact [communicationsupport@avalara.com](mailto:communicationsupport@avalara.com) in order to set up or establish the appropriate preferences.

### 1.12 Live vs. Test Transactions

Each client is provided with a three-character company code when signing up for the AFC SaaS Pro product. In order to run a live transaction, the client's company code must be placed in the

CompanyIdentifier field of the Transaction to be processed using AFC SaaS Pro (see [SalesUseTransaction](#) data definition). Only taxes for transactions that contain this code in the CompanyIdentifier field will be included in the monthly compliance files created by the AFC SaaS Pro Web Service. The CompanyIdentifier field may be left blank or use a value other than the assigned company code in order to process transactions where the taxes generated are to be excluded from the monthly compliance reports such as test transactions or quotes.

### 1.13 Compliance Month

Optional10 field applies when Date Method 2 is used. The month for the billing cycle must be specified in the Optional10 field of the transaction by entering the year and month in YYYYMM format. This will allow the transaction and taxes to be reported in the appropriate compliance reports for the specified month.

### 1.14 Tax Grouping

The following options determine how tax calculation results are returned after calling one of the tax calculation API functions. By changing this option, the taxes returned by the tax calculation API function may be grouped according to this setting.

**NOTE:** *This option will not modify the way that tax calculation results are logged into the AFC log data. 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.*

Please contact [communicationsupport@avalara.com](mailto:communicationsupport@avalara.com) in order to have tax grouping options set or activated.

#### 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.

#### Default Option:

- **groupresults=default (default)**  
The default option indicates that the taxes returned in the tax table after processing a transaction with AFC will not be grouped. Each tax will be returned in an individual tax record.

#### Tax Level Options:

- **groupsamelevel**  
This option will cause rate-based taxes at the same level to be grouped together. For example, if AFC returns a Local Sales Tax and a Local District Tax, these taxes will be grouped together since they have the same tax level (Local).
- **groupstate\_groupcountyandlocal**  
This option will cause all state taxes to be grouped together into a single record, and all county and local taxes to be grouped together into a separate record.
- **groupstatecountyandlocal**  
This option will cause all state, county, and local taxes to be grouped together into a single record.

#### Sales Tax Options:

The following options may be used in combination with any of the tax level options specified above. These options may be used to group sales taxes separately from other taxes. This option must be appended to the EZTax.cfg file following the line for the tax level group option.

- **groupsales**  
This option will group Sales Taxes (tax type 1) and Use Taxes (tax type 49) taxes into a separate record according to the tax level option being specified.
- **groupsalescategory**  
When using this option, AFC will group any items that are considered Sales Taxes together. In addition to the Sales Tax (tax type 1) and Use Tax (tax type 49), some District and Transit taxes are also in the sales tax category.

#### Examples

- **groupstatecountylocal**  
By entering the configuration option shown above into the EZTax.cfg file, all State, County, and Local taxes will be grouped together into a single record.
- **groupstatecountylocal**
- **groupsalescategory**

By entering these two options in the configuration file, all State, County, and Local sales category taxes will be grouped together into a single record. Any other State, County and Local taxes (if any) will be grouped into a separate record.

#### Tax Return Table

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

1. Jurisdiction Code. Jurisdiction Code (PCode) 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.
2. 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.
3. 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.
4. Tax amount. This field will contain the sum of the tax amount for all taxes being grouped together.
5. 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.

### 1.15 Specifying a Unique Identifier

AFC SaaS Pro does not provide in the response a unique identifier for a specific transaction. However, there are numerous reporting fields which can be used for the requesting application to populate a unique identifier. Reporting fields do not impact tax calculations. It is recommended that clients generate or construct the unique identifier in a way that is meaningful to their billing application and then proceed to provide that data to Avalara through one of the available reporting fields.

Reporting fields that can be used individually or in combination to represent a unique identifier are shown in the table below.

Reporting Fields	
Column Name	Description
Optional Optional4 Optional5 Optional6 Optional7 Optional8 Optional9 Optional10 InvoiceNumber ServiceLevelNumber	Unsigned int. Should be $\geq 0$ . The max value is different for different platforms, but in general supports values at least up to 4294967295 ( $2^{32} - 1$ ).
OptionalAlpha1 CustomerNumber	Alpha-numeric field. Supports up to 20 bytes.
OptionalFields	Alpha-numeric field. Supports up to 10 fields and up to 150 bytes per field.

## 1.16 Commit/Uncommit

The process to commit documents is optional and used to identify documents that should be included in the Compliance Reports for remittance to the Department of Revenue (DoR) for tax compliance reporting. This process is often used when not all taxes being calculated are considered final for compliance reporting. When a document is considered final, a commit can be sent to finalize the document. This also allows users to commit an entire invoice by using the document code instead of calculating the taxes again when the document is determined to be final.

**Note:** This feature is supported with the use of all tax calculations. Please reference the **AvaTax for Communications SaaS Pro Commit/Uncommit Process** document for additional information and details.

## 1.17 Application of Tax Brackets and Limits

### 1.17.1 Tax 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 SaaS Pro 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.

As an example of applying tax brackets, 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 result in  $(\$500 \times 2\%) = \$10$  plus  $(\$700 \times 1\%) = \$7$  which is a total tax of \$17.00.

### 1.17.2 Tax 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 SaaS Pro supports these transactions and the user is not required to make changes to account for it.

As an example of applying tax limits, 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.



## 2. Specifying a Tax Jurisdiction

For AFC SaaS Pro to calculate taxes for transactions correctly, it must first determine the taxing jurisdiction for the transaction in question. The tax laws of various jurisdictions complicate this. AFC SaaS Pro currently supports multiple unique rules for determination of the correct tax jurisdiction. Jurisdiction determination is usually based upon three inputs to AFC SaaS Pro:

1. The Ship From address.
2. The Ship To address.
3. The FOB point of the transaction (Shipping point or Destination).

Jurisdiction information can be supplied to the system in different ways. It can be supplied using a permanent jurisdiction code (PCode), using a FIPS Code, or using zip code and address information. AFC SaaS Pro allows the user to identify the jurisdictions by any of these methods, thereby providing maximum flexibility. AFC SaaS Pro allows the specification of jurisdiction information using any combination of these methods during the same session.

### PCode

PCodes are permanent jurisdiction codes that Avalara provides that allow AFC SaaS Pro users to populate their databases with jurisdiction information. With PCodes, Avalara clients can populate their customer records with jurisdiction information and never worry about changes of jurisdiction codes. If a jurisdiction code changes, Avalara re-maps the PCodes so clients are not effected. This Avalara's clients to populate client records with a PCode.

### FIPS Code

FIPS Codes are issued by the National Institute of Standards and Technology (NIST). AFC SaaS Pro provides internal translation tables from FIPS Codes to PCodes, so that using FIPS Codes is almost as fast and accurate as using PCodes. Some special taxing districts are not identified in separate FIPS codes, so there is some loss of accuracy, but the majority of transactions will produce the same tax results with FIP Codes as with PCodes.

### ZIP Code

The accuracy of the ZIP code method depends upon the amount of data provided for the address as well as the user's ability to choose the correct taxing jurisdiction zip code and address. AFC SaaS Pro databases contain numerous duplicate zip codes that cross not only taxing jurisdiction boundaries, but boundaries of localities as well. Providing a complete address along with the zip code insures the best match possible. When address information is missing, AFC SaaS Pro returns taxes based upon the first match of the provided input information.

It is appropriate to use the zip code interface or PCode methods for transactions where the jurisdiction can be positively identified by the calling application.

**Note:** For the United States, clients may enter various forms of the zip code plus 4 in the ZipCode and ZipP4 fields (with or without a dash or space). Please reference the table provided below for examples of the alternate formats which can be entered for US zip codes.

United States	
ZipCode Field	ZipP4 Field
12345	1234
12345 1234	
12345-1234	
123451234	

**Note:** For Canada, clients may populate the 6-digit Postal Code in the ZipCode field (with or without a dash or space). Canadian Postal Codes may also be entered by populating the first three in the ZipCode field and the last three in the Zip plus 4 (ZipP4) field.

For example, the Canadian zip code A1A 0A0 may be entered in any of the formats featured in the table below.

ZipCode Field	ZipP4 Field
A1A 0A0	
A1A-0A0	
A1A0A0	
A1A	0A0

## Tax Jurisdiction

AFC SaaS Pro provides the ability to obtain the PCode of the tax jurisdiction for a transaction by passing jurisdiction and transaction/service pair information.

**NOTE:** The transaction/service pair is required and must be provided in order to determine the correct tax jurisdiction.

## 2.1 Getting the Right Tax Jurisdiction for Local Taxation

It is important to get the end users location correct for local taxation. People tend to know what they are charged for local taxes. These taxes also have a tendency to change from one neighborhood to the next.

Avalara provides a comma delimited ASCII file to aid with entry of customers in your system and to help insure the correct taxing jurisdiction is setup. The file is "all\_adr.txt". This comma delimited ASCII files is provided to allow Avalara clients to populate database tables in their system with this information. The all address (all\_adr.txt) file is a cross-reference of locations to PCodes. The format of the files is illustrated below.

The "all\_adr.txt" file format and example follows:

**Pcode,P/A,Country,State,County,City,"Zip code range start","Zip code range end"**

```
3346800,0,USA,PA,LANCASTER,BART,17503,17503
3346900,0,USA,PA,LANCASTER,BAUSMAN,17504,17504
3347000,0,USA,PA,LANCASTER,BIRD IN HAND,17505,17505
3347100,0,USA,PA,LANCASTER,BOWMANSVILLE,17507,17507
3347200,0,USA,PA,LANCASTER,CHRISTIANA,17509,17509
3347300,0,USA,PA,LANCASTER,CHURCHTOWN,17555,17555
3347400,0,USA,PA,LANCASTER,CONESTOGA,17516,17516
3347500,0,USA,PA,LANCASTER,DRUMORE,17518,17518
3347600,0,USA,PA,LANCASTER,EAST EARL,17519,17519
3347700,0,USA,PA,LANCASTER,ELM,17521,17521
3347800,0,USA,PA,LANCASTER,GAP,17527,17527
```

## 3. Invoice Mode

### 3.1 Overview

AFC SaaS Pro 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. When in Invoice Mode, AFC SaaS Pro maintains a history of the transactions and applies the Steps, Brackets and/or Limits to the entire group of transactions.

AFC SaaS Pro provides the ability to process up to 30,000 transactions within an invoice for a single transaction or up to 50,000 transactions within an invoice for a batch transaction in Invoice Mode. Tax calculation results will be summarized by jurisdiction and tax type. Optionally, the individual taxes for each line item can also be returned in the output, but be advised that the response size may be up to a couple megabytes and may take up to one minute to process depending on the number of transactions in the customer batch. In addition, it is recommended to increase the timeout of your web API calls to 10 minutes although response times are expected to be much shorter even on the largest batches.

**NOTE:** The Invoice Mode feature must be individually activated for each client. Please contact [communicationsupport@avalara.com](mailto:communicationsupport@avalara.com) in order to access this functionality.

Also, it is recommended that transactions in Invoice Mode are contained within one monthly billing cycle.

### 3.2 Batch Transaction Submission

The CalcTaxesInCustMode APIs can be used for submitting a batch of telecom and/or sales and use transactions and adjustments within a single SOAP request in order to be processed using Invoice Mode. The steps for processing an invoice in using this method are the following:

1. Invoke the CalcTaxesInCustMode API to calculate taxes on all transactions that have been submitted. This API accepts a parameter indicating if the detailed taxes for each line item should be returned or if only the summarized taxes should be returned.

### 3.3 Single Transaction Submission

The following APIs can be used in order to submit each transaction within an individual SOAP request. The advantage of using this method is that each transaction can be validated prior to processing the entire batch. The steps for processing an invoice in using this method are the following:

1. Invoke the BeginCustomerBatch API to flag the beginning of a new customer transaction batch. AFC SaaS Pro will return a customer batch ID which will be used when submitting each transaction to the web service.
2. Submit each line item within the invoice by invoking one of the Invoice Mode APIs (CalcCustTaxes, CalcCustAdj, SAUCalcCustTaxes, or SAUCalcCustAdj) and pass the customer batch ID along with the telecom or sales and use transaction as parameters to the API call. The web service will not calculate taxes on the transaction at this point. It will simply keep track of each transaction in the batch.
3. Invoke the ProcessCustomerBatch API to calculate taxes on all transactions that have been submitted for that customer batch ID. This API accepts a parameter indicating if the detailed taxes for each line item should be returned or if only the summarized taxes should be returned.

After invoking ProcessCustomerBatch, the customer batch ID will no longer be valid.

**Note:** It is recommended that the batch transaction method be attempted first and that the single transaction method only be utilized if the batch transaction method does not work.

### 3.4 Taxing Jurisdiction Specification

Invoice Mode APIs accept either combination of PCodes, FIPS Codes, or Zip Address for the ship-to, ship-from. However, make sure that only one property is assigned for each location. For example, if setting a value in the ShipFromPCode field, leave the ShipFromFipsCode and ShipFromAddress fields null.

### 3.5 API Results

The response from any of these batch mode APIs contain two fields:

- **SummarizedTaxes.** This is an array of CustomerTaxData objects containing the accumulated taxes grouped by tax type and jurisdiction.
- **Taxes.** This is an array of TaxData objects containing the individual tax calculation results for each transaction in the batch. This array is populated only if the returnDetail parameter for the Invoice Mode API parameter is set to true. It is recommended to place a unique identifier within one of the optional fields of the transaction in order to be able to match the taxes in this array to its corresponding line item. (Please see [Specifying a Unique Identifier](#) for additional details).

### 3.6 Supported and Non-Supported Features

Not all APIs or options are supported in Invoice Mode. Invoice Mode only supports tax calculation methods. Please reference the table below for a high-level overview of supported features within this functionality.

Invoice Mode Supported Features
<b>Standard Tax Calculation Features Supported</b>
Tax calculations with FipsCode
Tax calculations with NpaNxx
Tax calculations with PCode
Tax calculations with ZipAddress
<b>Additional Features Supported</b>
Adjustments
Exclusions
Exemptions
Extended Optional Fields
Tax Inclusive*

*\*The Tax Inclusive feature is supported only through use of the Tax Inclusive flag. It is not supported through use of the API.*

Invoice Mode Non-Supported Features
<b>Additional Features Not Supported</b>
Overrides via APIs
Tax Inclusive APIs

### 3.7 Tax Inclusive Transactions

In order to process tax inclusive tax calculations in Invoice Mode, set the TaxInclusive property of the [SalesUseTransaction](#) field to true. AFC SaaS Pro will determine the appropriate base sale amount required to arrive at the total desired charge. The calculated base sale amount will be included in the TransCharge field of the corresponding [TaxDataV2](#) objects returned for that transaction.

### 3.7.1 Tax Inclusive - Batch Transaction Submission

The CalcTaxesInCustModeV2 APIs can be used for submitting a batch of telecom and/or sales and use transactions and adjustments within a single SOAP request in order to be processed using Invoice Mode. The steps for processing an invoice in using this method are the following:

1. Invoke the CalcTaxesInCustModeV2 API to calculate taxes on all transactions that have been submitted. This API accepts a parameter indicating if the detailed taxes for each line item should be returned or if only the summarized taxes should be returned.

### 3.7.2 Tax Inclusive - Single Transaction Submission

The following APIs can be used in order to submit each transaction within an individual SOAP request. The advantage of using this method is that each transaction can be validated prior to processing the entire batch. The steps for processing an invoice in using this method are the following:

4. Invoke the BeginCustomerBatch API to flag the beginning of a new customer transaction batch. AFC SaaS Pro will return a customer batch ID which will be used when submitting each transaction to the web service.
5. Submit each line item within the invoice by invoking one of the Invoice Mode APIs (CalcCustTaxes, CalcCustAdj, SAUCalcCustTaxes, or SAUCalcCustAdj) and pass the customer batch ID along with the telecom or sales and use transaction as parameters to the API call. The web service will not calculate taxes on the transaction at this point. It will simply keep track of each transaction in the batch.
6. Invoke the ProcessCustomerBatchV2 API to calculate taxes on all transactions that have been submitted for that customer batch ID. This API accepts a parameter indicating if the detailed taxes for each line item should be returned or if only the summarized taxes should be returned.

After invoking ProcessCustomerBatchV2, the customer batch ID will no longer be valid.

**Note:** It is recommended that the batch transaction method be attempted first and that the single transaction method only be utilized if the batch transaction method does not work.

## 4. Generating Custom Reports

### 4.1 Report Process Overview

AFC SaaS Pro provides the ability to dynamically generate a custom report containing the transactional data available in the database for a specified timeframe. The general process is as follows:

1. Invoke the [CreateReport](#) API in the web service. The [ReportOptions](#) parameter specifies which data to include in the report.
2. The report is processed in the background. The resulting output file is placed in the designated client FTP folder where it can be downloaded. This process takes anywhere from 30-90 minutes to complete.
3. Once the report has been generated and is ready for download, an email notification is sent to the email address(s) included in the [ReportOptions](#).

**Note:** Prior to the first time executing the CreateReport API call, please notify [communicationsupport@avalara.com](mailto:communicationsupport@avalara.com). If not already established, an FTP account needs to be created and the FTP credentials provided.

### 4.2 Setting up ReportOptions

#### 4.2.1 BaseReport

BaseReport specifies a report type to be used as a starting template for the customlog report. Additional columns can then be included or sorted upon using [CustomLogField](#). This field only needs to be set when the [ReportType](#) field is set to “customlog”. For further information, see [4.3 – Report Types](#) and [4.4 - Custom Log Report Columns](#).

The default value for this field is NULL.

#### 4.2.2 CreateNbaFile

If set to TRUE, all non-billable amounts returned as part of the report request are placed in a separate .nba file. If set to FALSE, the non-billable rows are returned as part of the requested report and no additional .nba file is created. The filename and file extension for the NBA file are specified in the FileName and FileExtension fields.

The default value for this field is 0.

### 4.2.3 CreateNcaFile

If set to TRUE, all non-compliance amounts returned as part of the report request are placed in a separate .nca file. If set to FALSE, the non-compliance rows are returned as part of the requested report and no additional .nca file is created. The filename and file extension for the NCA file are specified in the FileName and FileExtension fields.

The default value for this field is 0.

### 4.2.4 CustomLogFields

CustomLogFields is a list of CustomLogField. Specify which column should be used using the Column property, if the Column should be included on the report using the Include property, and if the Column should be used for sorting using the Sort property. For further information about the columns, refer to [4.4 - Custom Log Report Columns](#).

Setting Include to TRUE includes the column in the output while setting it to FALSE removes the specified column from the output. Setting Sort to TRUE uses the specified column as part of the sort on the report while setting it to FALSE does not use the column in the sort.

For example:

- Setting both Include and Sort to TRUE returns the specified column as part of the output as well as sorts the report using this column.
- Setting Include to FALSE and Sort to TRUE does not include the column in the output, but will sort the report based on this column.
- Setting Include to TRUE and Sort to FALSE includes the column in the output, but does not sort the report based on this column.

**Note:** The sorting of a custom report is based upon the order in which the columns are passed to the API.

**Note:** Only transactions and taxes that include the 3-character company code for your account in the CompanyIdentifier field of the input TelecomTransaction or SalesUseTransaction will be included in the report.

### 4.2.5 EmailAddress

Multiple email addresses may be included and should be separated by a semicolon and space (; ). When the report is available on the client FTP site, an email is sent to those addresses contained in the EmailAddress field notifying the recipient(s) that the file is ready for download.

This field is required.



#### 4.2.6 FileExtension

The FileExtension field specifies the file extension for the output file. The period for the extension is not required. For example, “csv”.

The default value for this field is csv.

#### 4.2.7 FileName

The FileName field of the [ReportOptions](#) parameter specifies the name of the output file.

This field allows a date-time format using the following identifiers:

Identifier	Description
yy	Last two digits of year
yyyy	4-digit year
MM	Month
dd	Day of month
hh	Hour
mm	Minutes
ss	Seconds

If using these identifiers in the file name, they must be included within curly braces.

For example, a value of “MyCustomReport{yyyyMMdd}” generates an output file name similar to “MyCustomReport20160101”.

The default value for this field is <ReportType>-yyyyMMddhhmmss.

#### 4.2.8 IncludeHeaders

Setting IncludeHeaders to TRUE includes the column headers as the first row of the returned report while setting the field to FALSE does not include a header row. This setting applies to the NBA and NCA files as well if generated.

The default value for this field is 0.

#### 4.2.9 Precision

The Precision field defines the number of places a decimal shall return.

The default value for this field is 0.

#### 4.2.10 ReportType

The ReportType field specifies whether to use a pre-defined Avalara report template or if the report shall be a custom report. For further information, see [4.3 – Report Types](#).

This field is required.

#### 4.2.11 StartDate, EndDate, and TimeSpan

The data to be included in the report is mainly specified by the StartDate and EndDate fields of the [ReportOptions](#) parameter that is passed in to the [CreateReport](#) API. The dates are compared against the server date at the time when the transaction was submitted and processed by the web service.

The StartDate may not be more than 90 days in the past. The EndDate or TimeSpan may not be more than 31 days apart from the StartDate. The EndDate is non-inclusive, so only records with a timestamp smaller than the EndDate will be included in the report. For example, to include all the data for January 2016 in the report, set the StartDate to “01/01/2016” and the EndDate to “02/01/2016”.

StartDate must be set but either EndDate or TimeSpan may be used. If both EndDate and TimeSpan are set, CreateReport uses the TimeSpan value.

StartDate is a required field. Either EndDate or TimeSpan is required.

### 4.3 Report Types

The following report types are available. Note that each of the reports specified below will be generated as CSV files.

Report Type	Description
srtcomma20l	Generates report output similar to the srtcomma20l utility in AFC License. The columns included are: CountryIso, State, County, Locality, TaxType, TaxLevel, Rate,

Report Type	Description
	<p>TaxAmount, SaleAmount, ExemptSaleAmount, RefundUncollect, NetTaxableMeasure, Minutes, and Lines.</p> <p>Please refer to <a href="#">TM_00523_srtcomma20l.pdf</a> for additional information.</p>
srtcomma20l-p	<p>Generates report output similar to the srtcomma20l utility in AFC License using the -p option to output PCodes instead of the location names. The columns included are: PCode, TaxType, TaxLevel, Rate, TaxAmount, SaleAmount, ExemptSaleAmount, RefundUncollect, NetSaleAmount, Minutes, and Lines.</p> <p>Please refer to <a href="#">TM_00523_srtcomma20l.pdf</a> for additional information.</p>
srtcomma20ld	<p>Generates report output similar to the srtcomma20ld utility in AFC License. The columns included are: CountryIso, State, County, Locality, TaxType, TaxDescription, TaxLevel, TaxLevelDesc, Rate, TaxAmount, GrossSales, ExemptSaleAmount, RefundUncollect, NetTaxableMeasure, Minutes, and Lines.</p> <p>Please refer to <a href="#">TM_00524_srtcomma20ld.pdf</a> for additional information.</p>
srtcommadetail	<p>Generates report output similar to the srtcommadetail utility in AFC License. The columns included are: CountryIso, State, County, Locality, TaxType, TaxLevel, DiscountType, CalcType, Rate, TaxAmount, GrossSales, ExemptSaleAmount, RefundUncollect, NetSaleAmount, Minutes.</p> <p>Please refer to <a href="#">TM_00525_srtcommadetail.pdf</a> for additional information.</p>
customlog	<p>A custom report containing the predefined columns of the report type specified in the BaseReport field of the <a href="#">ReportOptions</a> and/or the fields specified in the <a href="#">CustomLogField</a> array.</p>

#### 4.4 Custom Log Report Columns

The following columns are available when generating a report using the customlog ReportType. For each column, the Include and Sort properties must also be specified.

A Calculated column refers to data calculated within and returned from the AFC Engine.

Column Name	Description	Calculated
CountryISO	Three-character country code for taxing jurisdiction.	No
State	State abbreviation for taxing jurisdiction.	No
County	County name for taxing jurisdiction.	No
Locality	Locality name for taxing jurisdiction.	No

Column Name	Description	Calculated
CompanyIdentifier	Input entered in the CompanyIdentifier field of the TelecomTransaction or SalesUseTansaction.	No
CustomerNumber	Input entered in the CustomerNumber field of the TelecomTransaction or SalesUseTansaction.	No
Date	Input entered in the Date field of the TelecomTransaction or SalesUseTansaction.	No
DiscountType	Input entered in the DiscountType field of the TelecomTransaction or SalesUseTansaction.	No
DiscountTypeDesc	Description for DiscounType field.	No
ExemptionType	Input entered in the ExemptionType field of the TelecomTransaction or SalesUseTansaction.	No
ExemptionTypeDesc	Description for ExemptionType field.	No
InvoiceNumber	Input entered in the InvoiceNumber field of the TelecomTransaction or SalesUseTansaction.	No
Optional	Input entered in the Optional field of the TelecomTransaction or SalesUseTansaction.	No
Optional4	Input entered in the Optional4 field of the TelecomTransaction or SalesUseTansaction.	No
Optional5	Input entered in the Optional5field of the TelecomTransaction or SalesUseTansaction.	No
Optional6	Input entered in the Optional6 field of the TelecomTransaction or SalesUseTansaction.	No
Optional7	Input entered in the Optional7 field of the TelecomTransaction or SalesUseTansaction.	No
Optional8	Input entered in the Optional8 field of the TelecomTransaction or SalesUseTansaction.	No
Optional9	Input entered in the Optional9 field of the TelecomTransaction or SalesUseTansaction.	No
Optional10	Input entered in the Optional10 field of the TelecomTransaction or SalesUseTansaction.	No
OptionalAlpha1	Input entered in the OptionalAlpha1 field of the TelecomTransaction or SalesUseTansaction.	No
ServiceLevelNumber	Input entered in the ServiceLevelNumber field of the TelecomTransaction or SalesUseTansaction.	No
Billable	Billable flag for tax.	No
CalcType	Calculation type for tax.	No
CalcTypeDesc	Description for CalcType field.	No
CategoryID	Category ID for tax.	No
CategoryDesc	Category description for tax.	No
ExemptSaleAmount	Exempt sale amount from tax record.	No
Compliance	Compliance flag from tax record.	No
PCode	PCode for taxing jurisdiction	No
Rate	Rate for tax.	No
RefundUncollect	For adjustments, this is the refunded taxable measure from tax record.	Yes
ServiceType	Service Type used in the input transaction or derived out of the bundle if a bundle transaction was used.	No
Surcharge	FET taxable flag from tax record	No

Column Name	Description	Calculated
TaxableMeasure	Taxable measure used for calculating the tax.	Yes
SaleAmount	Taxable measure plus exempt sale amount (typically the input sale amount entered in the transaction. This field is used for srtcomma20l and srtcomma20l-p reports.	Yes
GrossSales	Gross sale amount. This field is used for srtcomma20ld and srtcommadetail reports.	Yes
NetSaleAmount	Net sale amount. This field is used for srtcomma20ld and srtcommadetail reports.	Yes
NetTaxableMeasure	Net taxable measure. This field is used for srtcomma20l and srtcomma20l-p reports.	Yes
TaxAmount	Tax amount.	Yes
TaxLevel	Tax level identifier.	No
TaxLevelDesc	Description for tax level identifier.	No
TaxType	Tax type identifier.	No
TaxDescription	Description for tax type.	No
TransactionID	Transaction ID used in the input transaction or derived out of the bundle if a bundle transaction was used.	No
ServiceID	Service ID used in the input transaction or derived out of the bundle if a bundle transaction was used.	No
Lines	Number of lines used for calculating the tax in per-line taxes.	Yes
Locations	Number of locations	Yes
Minutes	Number of minutes used for calculating the tax in per-minute taxes.	Yes
TaxLogID	Primary key identifier for transaction.	No
ServerDate	Server date when the transaction was processed.	No
OptionalField1	Additional Extended Optional Fields in which column headers are defined by the user upon creation of each one.	No
OptionalField2		
OptionalField3		
OptionalField4		
OptionalField5		
OptionalField6		
OptionalField7		
OptionalField8		
OptionalField9		
OptionalField10		

## 4.5 Aggregation

Any records included in the report will be grouped together using any non-calculated fields that are included in the output. The non-calculated field values are aggregated. To avoid any grouping, include the TaxLogID column in the report output. For a list of calculated and non-calculated fields, see [4.3 - Custom Log Report Columns](#).

## 4.6 Output Files

The output files are compressed into a zip file with the name as specified in the FileName field. This zip file is then placed in the specific client FTP folder. The zip file may contain multiple files if the CreateNbaFile or CreateNcaFile options are set to true in the [ReportOptions](#).

### 4.6.1 Accessing the FTP Site

AFC SaaS Pro provides two ways to connect to the AFC FTP server - secure file-transfer protocol (SFTP) or file-transfer protocol (FTP) connection through an existing account and typing 'ftp.billsoft.com' at the prompt within the FTP session or by connecting to the web interface at <https://ftp.billsoft.com>.

All client FTP sites must be set up properly and clients must have their username and password. Please contact [communicationsupport@avalara.com](mailto:communicationsupport@avalara.com) to obtain this information or for assistance with accessing output files.

#### **Note Concerning FTP Client Software**

Please be aware that web browsers (Microsoft Internet Explorer, Firefox, etc.) cannot be used for uploading and downloading files via FTP to the AFC FTP site. A FTP client application must be used to transfer files to and from the AFC FTP site when using our service as FTP.

#### **Windows users:**

Microsoft does not currently include convenient FTP client software in its operating systems. It is assumed you either own a third-party FTP client application such as WS\_FTP (a shareware version of which can be downloaded from <http://www.ipswitch.com>), or that you are comfortable accessing FTP sites using command-line syntax.

#### **Linux users:**

If using a Linux system to transfer your call-data files, you can use any number of free FTP clients to contact the AFC FTP site, such as WXFtp, or you can use command-line syntax.

##### 4.6.1.1. Logging on to the FTP Site

To log on to the AFC FTP site, you will need the following information:

**HOSTNAME/ADDRESS:** 'ftp.billsoft.com' within a FTP session or via the web at <https://ftp.billsoft.com>

**USER ID:** Your User ID is created and provided to you. Please contact [communicationsupport@avalara.com](mailto:communicationsupport@avalara.com) for your organization's assigned User ID.

**PASSWORD:** Your organization's login password is also provided by your Avalara technical support contact. For future reference, you may make note of your login details in spaced provided below.

User ID: \_\_\_\_\_

Password: \_\_\_\_\_

Company Code: \_\_\_\_\_

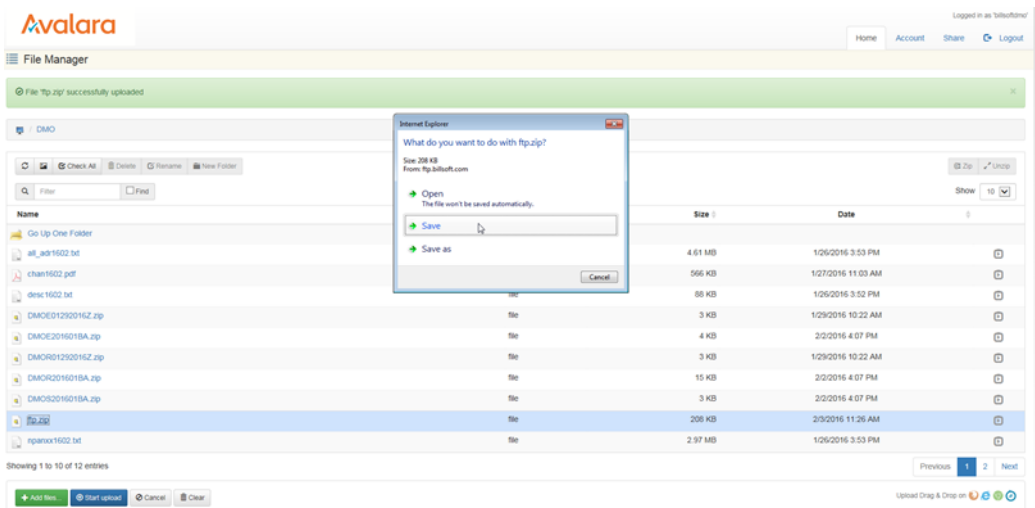
#### 4.6.1.2. Downloading Files from the FTP Site

Once you are connected to the AFC FTP site, you should change to the directory that has the same name as your company's three-character code.

When the report is available, you can either download the output from the AFC FTP or get it from the <https://ftp.billsoft.com> site.

To download from the AMC FTP site, connect within a FTP session, open the folder with you company's three-character code, and select the file you wish to download. Transfer the selected file to your destination using the tool's transfer method.

To download from the <https://ftp.billsoft.com> web interface, follow the steps below:

Step	Action/Result
1	View the completed files processed and locate the file with appropriate date and timestamp.
2	Highlight or double-click the file you wish to open. 
3	Select the <b>Save</b> or <b>Save as</b> option in the window.  <b>Note:</b> The file must be saved. If the file is opened and closed without being saved, it will no longer be available on the FTP site to download and save for your records.

## 4.7 Sample Code

```
// Create instance of web service client
var client = new EZTaxWebServiceClient("BasicHttpBinding_IEZTaxWebService");
client.ClientCredentials.UserName.UserName = "(MyUserName)";
client.ClientCredentials.UserName.Password = "(MyPassword)";

// Create a ReportOptions object to specify the options for the report
var reportOptions = new ReportOptions
{
    BaseReport = "srtcomma201",
    CreateNbaFile = true,
    CreateNcaFile = false,
    EmailAddress = "myemail@address.com",
    EndDate = new DateTime(2016, 5, 1),
    FileExtension = ".csv",
    FileName = "test-{yyyyMMdd}",
    IncludeHeaders = true,
    Precision = 6,
    ReportType = "customlog",
    StartDate = new DateTime(2016, 4, 1)
};

// For customlog reports, you may specify individual fields to be included
reportOptions.CustomLogFields = new[]
{
    new CustomLogField
    {
        Column = "PCode",
        Include = true,
        Sort = false
    },
    new CustomLogField
    {
        Column = "CategoryID",
        Include = true,
        Sort = true
    },
    new CustomLogField
    {
        Column = "CategoryDesc",
        Include = true,
        Sort = false
    }
};

try
{
    // Invoke the CreateReport API to submit the report for processing
    client.CreateReport(reportOptions);
    Console.WriteLine("Report submitted for processing.");
}
catch (FaultException ex)
{
    Console.WriteLine("An error occurred submitting the report: " + ex.ToString());
}
```



## 5. Optional Fields

### 5.1 Overview

AFC SaaS Pro provides a number of optional fields for use in reports.

There are two types of Optional Fields available in AFC SaaS Pro: 9 Default Optional Fields and up to 10 Extended Optional Fields. All Optional Fields are part of the [SalesUseTransaction](#).

**Note:** The Optional Fields do not impact taxation.

### 5.2 Default Optional Fields

There are 9 default Optional Fields.

Reporting Fields	
Column Name	Description
Optional	Unsigned int. Should be $\geq 0$ . The max value is different for different platforms, but in general supports values at least up to 4294967295 ( $2^{32} - 1$ ).
Optional4	
Optional5	
Optional6	
Optional7	
Optional8	
Optional9	
Optional10	
OptionalAlpha1	Alpha-numeric field. Supports up to 20 bytes.

**Note:** Please refer to Sections **1.10 Compliance Files** and **1.12 Compliance Month** for additional information regarding the Optional10 field.

### 5.3 Extended Optional Fields

AFC SaaS Pro has the ability to pass up to 10 additional Extended Optional Fields. The general process is as follows:

1. Add the Extended Optional Fields using the [OptionalFields](#) property of the [SalesUseTransaction](#).
2. Update the description of a specified Extended Optional Field by invoking the [UpdateOptionalFieldKeyDesc](#) API.
3. Display a list of all updated descriptions by invoking the [GetOptionalFieldKeyDesc](#) API.

The Extended Optional Fields are not currently supported in Single-Transaction Invoice Mode or Batch-Transaction Invoice Mode.

**Note:** The Extended Optional Fields passed as part of the Transaction are persisted until otherwise changed. For example, if the first [OptionalField](#) is used in a transaction and updated with a description of “Purchase Order Number”, this field should continue to be utilized as a Purchase Order Number until at least the end of the reporting cycle so that mixed data does not appear on the resulting Detail Log report or Custom Report.

### 5.3.1 Setting Up an Extended Optional Field in a Transaction

A Transaction containing one or more Extended Optional Fields is set up as usual with the addition of the [OptionalField](#) property. Each [OptionalField](#) specifies an OptionalKeyNo and an OptionalValue.

Field	Description
<b>OptionalKeyNo</b>	Optional field key number (integer values from 1 to 10). The OptionalKeyNo field is the key associated with the corresponding value.
<b>OptionalValue</b>	Value for optional field (up to 150 characters). The OptionalValue field is the value that will display in the body of the reports.

**Note:** To avoid possible issues with CSV reports, it is advised that the use of special characters, especially the comma (,) should be avoided.

### 5.3.2 UpdateOptionalFieldKeyDesc

The [UpdateOptionalFieldKeyDesc](#) API is used to update the description for each optional field key thereby indicating what each field in the [OptionalField](#) array of the [SalesUseTransaction](#) is used for (e.g. invoice number, line item number, etc.). The values specified will be used when generating reports that contain these optional fields.

To update a description, pass the OptionalKeyNo that should be updated as well as the desired description in the OptionalValue field.

Field	Description
<b>OptionalKeyNo</b>	Optional field key number (integer values from 1 to 10). The OptionalKeyNo field is the key originally passed as part of the transaction.
<b>OptionalValue</b>	Value for the optional field description (up to 150 characters). The OptionalValue field is the value of the description of the specified Optional Field that will display on the reports.

**Note:** To avoid possible issues with CSV reports, it is advised that the use of special characters, especially the comma (,) should be avoided.

### 5.3.3 GetOptionalFieldKeyDesc

The [GetOptionalFieldKeyDesc](#) API returns the description for each optional field key in the [OptionalField](#) array of the [SalesUseTransaction](#) that has been updated using the [UpdateOptionalFieldKeyDesc](#) API. The description values specified will be used when generating reports that contain these optional fields.

## 6. Zip Lookup Requests

### 6.1 Overview

Zip Lookup functionality returns all or multiple PCodes and jurisdiction details associated with the address input provided.

To use this functionality, invoke the [ZipLookup](#) API.

### 6.2 Setting up a Request

In order to call the API, the ZipLookup data structure needs to be populated. At least one location field is required. If Country is not specified, the default value is set to “USA”.

#### 6.2.1 Best Match

By default the ZipLookup algorithm requires an exact match of all fields provided with the following considerations:

1. Punctuation in city names will be ignored
2. Whitespace in city names will be ignored. For example, the city name LAND O' LAKES will match LAND O LAKES or LANDOLAKES as both the punctuation and spacing will be ignored for purposes of matching.

If Best Match is set to TRUE, some limited variable search algorithms will be used to find the best match for the data provided.

Setting the **BestMatchFlag** to TRUE

- Best Match first attempts an exact match
- If and only if the exact match search fails, the best match algorithm will be applied to find matches based on variable search algorithms.

Setting the **BestMatchFlag** to FALSE produces exact matches only.

- Produces exact matches only

The results field **MatchTypeApplied** indicates whether Exact match or Best match was used.

### 6.2.2 Location Data

At least one location field is required. If Country is not specified, the default value is set to "USA".

Specify a location by providing one or more of the following fields: **City, Country, County, State, ZipCode**.

### 6.2.3 Limit Results

The **LimitResults** field is used to limit the number of matches returned.

The default value is 100 while the maximum limit is 1,000. If 0 is entered as a value, the default limit is used. If the value entered is greater than the maximum limit, the maximum limit is used.

## 6.3 Results

The [LocationData](#) structure is returned for each location found based on the provided input. All results are contained within the [ZipLookupResult](#) structure.

## 6.4 Examples

Examples using Manhattan in New York are as follows:

### 1. Sample Request with one exact match

**Request 1 - Match exact on Country, State, County, City and Zip Code**

Country	State	County	Locality	ZipCode	Best Match
USA	NY	New York	Manhattan	10001	FALSE

**Result**

PCode	Country	State	County	City
2604100	USA	NY	NEW YORK	MANHATTAN

## 2. Sample Request with multiple exact matches

### Request 1 – Match exact on Country, State, County and Zip Code

Country	State	County	Locality	ZipCode	Best Match
USA	NY	New York		10001	FALSE

### Request 2 – Match exact on Country and Zip Code (Blank Country defaults to USA)

Country	State	County	Locality	ZipCode	Best Match
				10001	FALSE

### Result - Both match 5 addresses (All have same Country, State, County and Zip)

PCode	Country	State	County	City
2604100	USA	NY	NEW YORK	EMPIRE STATE
2604100	USA	NY	NEW YORK	GREELEY SQUARE
2604100	USA	NY	NEW YORK	ONE HUNDRED THIRTY EIGHTH
2604100	USA	NY	NEW YORK	MANHATTAN
2604100	USA	NY	NEW YORK	NEW YORK

## 3. Sample Request – Match exact on invalid address

### Request 1 – Match exact on Country, State, County, City and Zip Code

Country	State	County	Locality	ZipCode	Best Match
USA	NY	Westchester	Manhattan	10001	FALSE

### Result – No matches, not a valid address

PCode	Country	State	County	City

## 4. Sample Request – Best match on invalid address

### Request 1 – Best match on Country, State, County, City and Zip Code

Country	State	County	Locality	ZipCode	Best Match
USA	NY	Westchester	Manhattan	10001	FALSE

### Result – County is disregarded

PCode	Country	State	County	City
2604100	USA	NY	NEW YORK	MANHATTAN

## 7. AFC SaaS Pro Sales and Use Web Service Programmer Reference

The AFC SaaS Pro Sales & Use Web Service was developed using XML, SOAP 1.1, and WSDL so it can be integrated into virtually any application. To use the web service from your application, you will need to create a proxy stub for your programming language and platform. The proxy stub will encapsulate many of the details of communicating over the Internet between your application and the web service. The proxy stub will contain data types, classes and functions that you will use in your source code to invoke the methods on the web service.

Most programming languages have a toolkit or SDK for generating some or all of the proxy stub. The following is a list of some of the products that may be used to create the proxy stub.

Toolkit or Product Name	Programming Language	Platform
Visual Studio.NET	C# or Visual Basic.NET	Microsoft Windows
.NET Framework SDK	C# or Visual Basic.NET	Microsoft Windows
Systinet Server	Java or C++	See <a href="http://www.systinet.com">http://www.systinet.com</a>
Apache Axis	Java or C++	See <a href="http://ws.apache.org/axis/">http://ws.apache.org/axis/</a>
GSOAP	C/C++	See <a href="http://sourceforge.net/projects/gsoap2">http://sourceforge.net/projects/gsoap2</a>

### 7.1 TaxService Endpoint

[TaxService WSDL](#) (secured with user id and password in SOAP header) Avalara recommends using HTTPS when connecting to this endpoint. The URL for this endpoint is <http://EZtaxasp.billsoft.com/EZtaxWebService/EZtaxWebService.svc?wsdl>.

There will be two ways to access the service, either via https or http. The TaxService endpoint requires the user id and password to be passed in a custom SOAP header element. These are sent in clear text, so Avalara requires the client to use one of the security models to secure the password.

#### Common Properties

WSDL URL <https://eztaxasp.billsoft.com/EZTaxWebService/EZTaxWebService.svc?wsdl>

Namespace <http://tempuri.org/>

#### Binding Specific Properties

Binding	Endpoint	Soap Version
BasicHttpBinding_IETaxWebService	<a href="https://eztaxasp.billsoft.com/EZTaxWebService/EZTaxWebService.svc">https://eztaxasp.billsoft.com/EZTaxWebService/EZTaxWebService.svc</a>	SOAP 1.1
BasicHttpBinding_IETaxWebService1	<a href="https://eztaxasp.billsoft.com/EZTaxWebService/EZTaxWebService.svc/Soap11">https://eztaxasp.billsoft.com/EZTaxWebService/EZTaxWebService.svc/Soap11</a>	SOAP 1.1
CustomBinding_IETaxWebService	<a href="https://eztaxasp.billsoft.com/EZTaxWebService/EZTaxWebService.svc/SSL">https://eztaxasp.billsoft.com/EZTaxWebService/EZTaxWebService.svc/SSL</a>	SOAP 1.2

## Deprecated Binding

Binding	Endpoint	Soap Version
WSHttpBinding_IETaxWebService	<a href="http://eztaxasp.billsoft.com/EZTaxWebService/EZTaxWebService.svc">http://eztaxasp.billsoft.com/EZTaxWebService/EZTaxWebService.svc</a>	SOAP 1.2

The remainder of this document describes the web service methods and customer data types required to call AFC SaaS Pro from your application. With this interface, the applications pass the transaction data to AFC SaaS Pro as they are being sold or billed. AFC SaaS Pro calculates all required taxes and returns the tax information to the billing system per transaction. In addition, AFC SaaS Pro stores all tax data generated in a database. This data can then be used to generate reports used for tax compliance. AFC SaaS Pro is capable of generating files that can be electronically processed by Atlantax, Ernst & Young, and Tax Partners, Inc., for tax compliance filing. AFC SaaS Pro provides facilities to generate tax adjustments or refunds based upon un-collectable accounts or customer refunds. Adjustment information is returned to the application and is utilized to update tax data for report generation and compliance filing.

## 8. Sales and Use Web Methods

These methods process each transaction independently. See Section 6 for the Invoice Mode web methods.

Any of these messages can produce an exception. Errors can occur in any of the calculation methods are listed in [Error Messages Common to all Calculation Methods](#). Messages that are specific to a particular method are shown with each method description.

Method Name	Summary
<a href="#">SAUCalcTaxesWithPCode</a>	Calculate taxes on supplied transaction using PCodes
<a href="#">SAUCalcTaxesWithFipsCode</a>	Calculate taxes on supplied transaction using FIPS Codes
<a href="#">SAUCalcTaxesWithZipAddress</a>	Calculate taxes on supplied transaction using ZIP Addresses
<a href="#">SAUCalcAdjWithPCode</a>	Calculate adjustment on supplied transaction using PCodes
<a href="#">SAUCalcAdjWithFipsCode</a>	Calculate adjustment on supplied transaction using FIPS Codes
<a href="#">SAUCalcAdjWithZipAddress</a>	Calculate adjustment on supplied transaction using ZIP Addresses
<a href="#">SAUCalcReverseTaxesWithPCode*</a>	Performs a tax inclusive calculation to arrive at the base sale amount and taxes for the desired total charge using PCodes.
<a href="#">SAUCalcReverseTaxesWithFipsCode*</a>	Performs a tax inclusive calculation to arrive at the base sale amount and taxes for the desired total charge using Fips Codes.
<a href="#">SAUCalcReverseTaxesWithZipAddress*</a>	Performs a tax inclusive calculation to arrive at the base sale amount and taxes for the desired total charge using Zip Address.

Method Name	Summary
<a href="#"><u>SAUCalcReverseAdjWithPCode*</u></a>	Calculates a tax inclusive adjustment to arrive at the base sale amount and taxes for the desired total charge using PCodes.
<a href="#"><u>SAUCalcReverseAdjWithFipsCode*</u></a>	Calculates a tax inclusive adjustment to arrive at the base sale amount and taxes for the desired total charge using Fips Codes.
<a href="#"><u>SAUCalcReverseAdjWithZipAddress*</u></a>	Calculates a tax inclusive adjustment to arrive at the base sale amount and taxes for the desired total charge using Zip Address.
<a href="#"><u>SAUCalcTaxInclusiveTaxesWithPCode</u></a>	Performs a tax inclusive calculation to arrive at the base sale amount and taxes for the desired total charge using PCodes.
<a href="#"><u>SAUCalcTaxInclusiveTaxesWithFipsCode</u></a>	Performs a tax inclusive calculation to arrive at the base sale amount and taxes for the desired total charge using Fips Codes.
<a href="#"><u>SAUCalcTaxInclusiveTaxesWithZipAddress</u></a>	Performs a tax inclusive calculation to arrive at the base sale amount and taxes for the desired total charge using Zip Address.
<a href="#"><u>SAUCalcTaxInclusiveAdjWithPCode</u></a>	Calculates a tax inclusive adjustment to arrive at the base sale amount and taxes for the desired total charge using PCodes.
<a href="#"><u>SAUCalcTaxInclusiveAdjWithFipsCode</u></a>	Calculates a tax inclusive adjustment to arrive at the base sale amount and taxes for the desired total charge using Fips Codes.
<a href="#"><u>SAUCalcTaxInclusiveAdjWithZipAddress</u></a>	Calculates a tax inclusive adjustment to arrive at the base sale amount and taxes for the desired total charge using Zip Address.
<a href="#"><u>BeginCustomerBatch</u></a>	Flags the beginning of a customer transaction batch and returns a customer batch ID that can be used to submit each transaction and process the entire batch.
<a href="#"><u>SAUCalcCustTaxes</u></a>	Submits a transaction for the specified customer batch.
<a href="#"><u>SAUCalcCustAdj</u></a>	Submits an adjustment for the specified customer batch.
<a href="#"><u>ProcessCustomerBatch</u></a>	Processes a customer batch and returns the tax calculation results.
<a href="#"><u>CalcTaxesInCustMode</u></a>	Processes a batch of telecom and/or sales and use transactions and adjustments in Invoice Mode.
<a href="#"><u>SAUCalcJurisdiction</u></a>	Determines the taxing jurisdiction for a transaction and returns the PCode at the lowest jurisdiction level.
<a href="#"><u>SAUCalcTaxesWithOverrides</u></a>	Accepts transaction data and tax rate override data to perform tax calculations with provided override information.
<a href="#"><u>SAUCalcAdjWithOverrides</u></a>	Accepts transaction data and tax rate override data to perform tax adjustment calculations with provided override information.
<a href="#"><u>CommitTransactions</u></a>	Used to commit or uncommit a DocumentCode.

*\*It is recommended to begin using the newly renamed 'tax inclusive' APIs as the 'reverse' APIs will be deprecated in the future.*

## 8.1 SAUCalcTaxesWithPCode

This method accepts transaction data and performs appropriate tax calculations. Ship from and ship to information is passed using PCodes.

*Return Type:*



[TaxData\[\]](#) An array of [TaxData](#) objects that contain the information about the taxes applied.

*Parameters:*

[SalesUseTransaction](#)

Sales & Use transaction data

[Nexus \[\]](#)

An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion \[\]](#)

An array of [Exclusion](#) objects. Can be null or an empty list.

*Errors:*

- ShipToPCode must be set! – The ShipToPCode property of the SalesUseTransaction was not set.
- ShipFromPCode must be set! – The ShipFromPCode property of the SalesUseTransaction was not set.
- Transaction is null! – No Transaction passed in
- PCode not found. – One of the two PCode values passed in is not in AFC

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:* This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

## 8.2 SAUCalcTaxesWithFipsCode

This method accepts transaction data and performs appropriate tax calculations. Ship from and ship to information is passed using FIPS Codes.

*Return Type:*

[TaxData\[\]](#) An array of [TaxData](#) objects that contain the information about the taxes applied.

*Parameters:*

[SalesUseTransaction](#)

Sales & Use transaction data

[Nexus \[\]](#)

An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion \[\]](#)

An array of [Exclusion](#) objects. Can be null or an empty list.

*Errors:*

- ShipToFipsCode must be set! – The ShipToFipsCode property of the SalesUseTransaction was not set.
- ShipFromFipsCode must be set! – The ShipFromPCode property of the SalesUseTransaction was not set.
- Transaction is null! – No Transaction passed in
- FIPS not found. – One of the two FipsCode values passed in is not in AFC
- FIPS (to PCode) cross-reference database not open. - – Indicates an error occurred on the server

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

### 8.3 SAUCalcTaxesWithZipAddress

This method accepts transaction data and performs appropriate tax calculations. Ship from and ship to information is passed using [ZipAddress](#) objects.

*Return Type:*

[TaxData](#)[] An array of [TaxData](#) objects that contain the information about the taxes applied.

*Parameters:*

[SalesUseTransaction](#) Sales & Use transaction data  
[Nexus](#) [] An array of [Nexus](#) objects. Can be null or an empty list.  
[Exclusion](#) [] An array of [Exclusion](#) objects. Can be null or an empty list.

*Errors:*

- ShipToZipAddress must be set! – The ShipToZipAddress property of the SalesUseTransaction was not set.
- ShipFromZipAddress must be set! – The ShipFromZipAddress property of the SalesUseTransaction was not set.
- Transaction is null! – No Transaction passed in
- County/State/Zip not found. – One of the two ZipAddress values passed in is not in AFC

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

### 8.4 SAUCalcAdjWithPCode

This method accepts transaction data and performs an appropriate tax adjustment. Ship from and ship to information is passed using PCodes.

*Return Type:*

[TaxData](#)[] An array of [TaxData](#) objects that contain the information about the taxes applied.

*Parameters:*

[SalesUseTransaction](#) Sales & Use transaction data  
[Nexus](#) [] An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion](#) [] An array of [Exclusion](#) objects. Can be null or an empty list.

*Errors:*

- ShipToPCode must be set! – The ShipToPCode property of the SalesUseTransaction was not set.
- ShipFromPCode must be set! – The ShipFromPCode property of the SalesUseTransaction was not set.
- Adjustment is null! – No Adjustment passed in
- PCode not found. – One of the two PCode values passed in is not in AFC

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

Adjustment methods are used to process refunds or credits. They rely on the [AdjustmentMethod field](#).

Discounts should be processed using the [DiscountAttributes](#) object in one of the tax calculation methods, not by calling the adjustment methods.

## 8.5 SAUCalcAdjWithFipsCode

This method accepts transaction data and performs an appropriate tax adjustment. Ship from and ship to information is passed using FIPS Codes.

*Return Type:*

[TaxData](#)[] An array of [TaxData](#) objects that contain the information about the taxes applied.

*Parameters:*

[SalesUseTransaction](#) Sales & Use transaction data

[Nexus](#) [] An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion](#) [] An array of [Exclusion](#) objects. Can be null or an empty list.

*Errors:*

- ShipToFipsCode must be set! – The ShipToFipsCode property of the SalesUseTransaction was not set.
- ShipFromFipsCode must be set! – The ShipFromPCode property of the SalesUseTransaction was not set.
- Adjustment is null! – No Adjustment passed in
- FIPS not found. – One of the two FipsCode values passed in is not in AFC
- FIPS (to PCode) cross-reference database not open. - – Indicates an error occurred on the server

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

Adjustment methods are used to process refunds or credits. They rely on the [AdjustmentMethod field](#).

Discounts should be processed using the [DiscountAttributes](#) object in one of the tax calculation methods, not by calling the adjustment methods.

## 8.6 SAUCalcAdjWithZipAddress

This method accepts transaction data and performs an appropriate tax adjustment. Ship from and ship to information is passed using [ZipAddress](#) objects.

### Return Type:

[TaxData](#)[] An array of [TaxData](#) objects that contain the information about the taxes applied.

### Parameters:

<a href="#">SalesUseTransaction</a>	Sales & Use transaction data
<a href="#">Nexus</a> []	An array of <a href="#">Nexus</a> objects. Can be null or an empty list.
<a href="#">Exclusion</a> []	An array of <a href="#">Exclusion</a> objects. Can be null or an empty list.

### Errors:

- ShipToZipAddress must be set! – The ShipToZipAddress property of the SalesUseTransaction was not set.
- ShipFromZipAddress must be set! – The ShipFromZipAddress property of the SalesUseTransaction was not set.
- Adjustment is null! – No Adjustment passed in
- County/State/Zip not found. – One of the two ZipAddress values passed in is not in AFC

See also [Error Messages Common to all Calculation Methods](#)

### Remarks:

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

Adjustment methods are used to process refunds or credits. They rely on the [AdjustmentMethod field](#).

Discounts should be processed using the [DiscountAttributes](#) object in one of the tax calculation methods, not by calling the adjustment methods.

## 8.7 SAUCalcReverseTaxesWithPCode

*\*It is recommended to begin using the newly renamed 'tax inclusive' API, [SAUCalcTaxInclusiveTaxesWithPCode](#), as the 'reverse' API will be deprecated in the future.*

This method accepts transaction data and performs a tax inclusive calculation in order to arrive at the base sale amount and taxes for the desired total charge. Ship from and ship to information is passed using PCodes.

*Return Type:*

[ReverseTaxResults](#)

Contains the calculated base sale amount for the transaction and an array of [TaxData](#) objects that contain the information about the taxes applied.

*Parameters:*

[SalesUseTransaction](#)

Sales & Use transaction data

[Nexus](#) []

An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion](#) []

An array of [Exclusion](#) objects. Can be null or an empty list.

*Errors:*

- ShipToPCode must be set! – The ShipToPCode property of the SalesUseTransaction was not set.
- ShipFromPCode must be set! – The ShipFromPCode property of the SalesUseTransaction was not set.
- Transaction is null! – No Transaction passed in
- PCode not found. – One of the two PCode values passed in is not in AFC

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

The desired total charge must be placed in the Charge field of the [SalesUseTransaction](#).

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

## 8.8 SAUCalcReverseTaxesWithFipsCode

*\*It is recommended to begin using the newly renamed 'tax inclusive' API, [SAUCalcTaxInclusiveTaxesWithFipsCode](#), as the 'reverse' API will be deprecated in the future.*

This method accepts transaction data and performs a tax inclusive calculation in order to arrive at the base sale amount and taxes for the desired total charge. Ship from and ship to information is passed using Fips Codes.

*Return Type:*

[ReverseTaxResults](#)

Contains the calculated base sale amount for the transaction and an array of [TaxData](#) objects that contain the information about the taxes applied.

*Parameters:*

[SalesUseTransaction](#)

Sales & Use transaction data

[Nexus](#) []

An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion](#) []

An array of [Exclusion](#) objects. Can be null or an empty list.

*Errors:*

- ShipToFipsCode must be set! – The ShipToFipsCode property of the SalesUseTransaction was not set.
- ShipFromFipsCode must be set! – The ShipFromPCode property of the SalesUseTransaction was not set.
- Transaction is null! – No Transaction passed in
- Fips Code not found. – One of the two Fips Code values passed in is not in AFC
- FIPS (to PCode) cross-reference database not open. – Indicates an error occurred on the server

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

The desired total charge must be placed in the Charge field of the [SalesUseTransaction](#).

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

## 8.9 SAUCalcReverseTaxesWithZipAddress

*\*It is recommended to begin using the newly renamed ‘tax inclusive’ API, [SAUCalcReverseTaxesWithZipAddress](#), as the ‘reverse’ API will be deprecated in the future.*

This method accepts transaction data and performs a tax inclusive calculation in order to arrive at the base sale amount and taxes for the desired total charge. Ship from and ship to information is passed using [ZipAddress](#) objects.

*Return Type:*

[ReverseTaxResults](#)

Contains the calculated base sale amount for the transaction and an array of [TaxData](#) objects that contain the information about the taxes applied.

*Parameters:*

[SalesUseTransaction](#)

Sales & Use transaction data

[Nexus](#) []

An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion](#) []

An array of [Exclusion](#) objects. Can be null or an empty list.

*Errors:*

- ShipToZipAddress must be set! – The ShipToZipAddress property of the SalesUseTransaction was not set.
- ShipFromZipAddress must be set! – The ShipFromZipAddress property of the SalesUseTransaction was not set.
- Transaction is null! – No Transaction passed in
- County/State/Zip not found. – One of the two ZipAddress values passed in is not in AFC

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

The desired total charge must be placed in the Charge field of the [SalesUseTransaction](#).

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

## 8.10 SAUCalcReverseAdjWithPCode

*\*It is recommended to begin using the newly renamed ‘tax inclusive’ API, [SAUCalcTaxInclusiveAdjWithPCode](#), as the ‘reverse’ API will be deprecated in the future.*

This method accepts transaction data and performs tax inclusive adjustments in order to arrive at the base sale amount and taxes for the desired total charge. Ship from and ship to information is passed using PCodes.

*Return Type:*

[ReverseTaxResults](#)

Contains the calculated base sale amount for the transaction and an array of [TaxData](#) objects that contain the information about the taxes applied.

*Parameters:*

[SalesUseTransaction](#)

Sales & Use transaction data

[Nexus](#) []

An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion](#) []

An array of [Exclusion](#) objects. Can be null or an empty list.

*Errors:*

- ShipToPCode must be set! – The ShipToPCode property of the SalesUseTransaction was not set.
- ShipFromPCode must be set! – The ShipFromPCode property of the SalesUseTransaction was not set.
- Adjustment is null! – No Transaction passed in
- PCode not found. – One of the two PCode values passed in is not in AFC

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

The desired total charge must be placed in the Charge field of the [SalesUseTransaction](#) as a positive number.

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

Adjustment methods are used to process refunds or credits. They rely on the [AdjustmentMethod field](#).

Discounts should be processed using the [DiscountAttributes](#) object in one of the tax calculation methods, not by calling the adjustment methods.

## 8.11 SAUCalcReverseAdjWithFipsCode

*\*It is recommended to begin using the newly renamed 'tax inclusive' API, [SAUCalcTaxInclusiveAdjWithFipsCode](#), as the 'reverse' API will be deprecated in the future.*

This method accepts transaction data and performs tax inclusive adjustments in order to arrive at the base sale amount and taxes for the desired total charge. Ship from and ship to information is passed using Fips Codes.

### Return Type:

[ReverseTaxResults](#)

Contains the calculated base sale amount for the transaction and an array of [TaxData](#) objects that contain the information about the taxes applied.

### Parameters:

[SalesUseTransaction](#)

Sales & Use transaction data

[Nexus](#) []

An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion](#) []

An array of [Exclusion](#) objects. Can be null or an empty list.

### Errors:

- ShipToFipsCode must be set! – The ShipToFipsCode property of the SalesUseTransaction was not set.
- ShipFromFipsCode must be set! – The ShipFromPCode property of the SalesUseTransaction was not set.
- Adjustment is null! – No Transaction passed in
- Fips Code not found. – One of the two Fips Code values passed in is not in AFC
- FIPS (to PCode) cross-reference database not open. - – Indicates an error occurred on the server

See also [Error Messages Common to all Calculation Methods](#)

### Remarks:



The desired total charge must be placed in the Charge field of the [SalesUseTransaction](#) as a positive number.

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

Adjustment methods are used to process refunds or credits. They rely on the [AdjustmentMethod field](#).

Discounts should be processed using the [DiscountAttributes](#) object in one of the tax calculation methods, not by calling the adjustment methods.

## 8.12 SAUCalcReverseAdjWithZipAddress

*\*It is recommended to begin using the newly renamed 'tax inclusive' API, [SAUCalcTaxInclusiveAdjWithZipAddress](#), as the 'reverse' API will be deprecated in the future.*

This method accepts transaction data and performs tax inclusive adjustments in order to arrive at the base sale amount and taxes for the desired total charge. Ship from and ship to information is passed using [ZipAddress](#) objects.

### Return Type:

[ReverseTaxResults](#)

Contains the calculated base sale amount for the transaction and an array of [TaxData](#) objects that contain the information about the taxes applied.

### Parameters:

[SalesUseTransaction](#)

Sales & Use transaction data

[Nexus](#) []

An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion](#) []

An array of [Exclusion](#) objects. Can be null or an empty list.

### Errors:

- ShipToZipAddress must be set! – The ShipToZipAddress property of the SalesUseTransaction was not set.
- ShipFromZipAddress must be set! – The ShipFromZipAddress property of the SalesUseTransaction was not set.
- Adjustment is null! – No Adjustment passed in
- County/State/Zip not found. – One of the two ZipAddress values passed in is not in AFC

See also [Error Messages Common to all Calculation Methods](#)

### Remarks:

The desired total charge must be placed in the Charge field of the [SalesUseTransaction](#) as a positive number.

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

Adjustment methods are used to process refunds or credits. They rely on the [AdjustmentMethod](#) field.

Discounts should be processed using the [DiscountAttributes](#) object in one of the tax calculation methods, not by calling the adjustment methods.

### 8.13 SAUCalcTaxInclusiveTaxesWithPCode

This method accepts transaction data and performs a tax inclusive calculation in order to arrive at the base sale amount and taxes for the desired total charge. Ship from and ship to information is passed using PCodes.

*Return Type:*

[TaxInclusiveTaxResults](#)

Contains the calculated base sale amount for the transaction and an array of [TaxData](#) objects that contain the information about the taxes applied.

*Parameters:*

[SalesUseTransaction](#)

Sales & Use transaction data

[Nexus](#) []

An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion](#) []

An array of [Exclusion](#) objects. Can be null or an empty list.

*Errors:*

- ShipToPCode must be set! – The ShipToPCode property of the SalesUseTransaction was not set.
- ShipFromPCode must be set! – The ShipFromPCode property of the SalesUseTransaction was not set.
- Transaction is null! – No Transaction passed in
- PCode not found. – One of the two PCode values passed in is not in AFC

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

The desired total charge must be placed in the Charge field of the [SalesUseTransaction](#).

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

### 8.14 SAUCalcTaxInclusiveTaxesWithFipsCode

This method accepts transaction data and performs a tax inclusive calculation in order to arrive at the base sale amount and taxes for the desired total charge. Ship from and ship to information is passed using Fips Codes.

*Return Type:*

[TaxInclusiveTaxResults](#)

Contains the calculated base sale amount for the transaction and an array of [TaxData](#) objects that contain the information about the taxes applied.

*Parameters:*

[SalesUseTransaction](#)

Sales & Use transaction data

[Nexus](#) []

An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion](#) []

An array of [Exclusion](#) objects. Can be null or an empty list.

*Errors:*

- ShipToFipsCode must be set! – The ShipToFipsCode property of the SalesUseTransaction was not set.
- ShipFromFipsCode must be set! – The ShipFromPCode property of the SalesUseTransaction was not set.
- Transaction is null! – No Transaction passed in
- Fips Code not found. – One of the two Fips Code values passed in is not in AFC
- FIPS (to PCode) cross-reference database not open. - – Indicates an error occurred on the server

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

The desired total charge must be placed in the Charge field of the [SalesUseTransaction](#).

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

## 8.15 SAUCalcTaxInclusiveTaxesWithZipAddress

This method accepts transaction data and performs a tax inclusive calculation in order to arrive at the base sale amount and taxes for the desired total charge. Ship from and ship to information is passed using [ZipAddress](#) objects.

*Return Type:*

[TaxInclusiveTaxResults](#)

Contains the calculated base sale amount for the transaction and an array of [TaxData](#) objects that contain the information about the taxes applied.

*Parameters:*

[SalesUseTransaction](#)

Sales & Use transaction data

[Nexus](#) []

An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion](#) []

An array of [Exclusion](#) objects. Can be null or an empty list.

*Errors:*

- ShipToZipAddress must be set! – The ShipToZipAddress property of the SalesUseTransaction was not set.

- ShipFromZipAddress must be set! – The ShipFromZipAddress property of the SalesUseTransaction was not set.
- Transaction is null! – No Transaction passed in
- County/State/Zip not found. – One of the two ZipAddress values passed in is not in AFC

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

The desired total charge must be placed in the Charge field of the [SalesUseTransaction](#).

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

## 8.16 SAUCalcTaxInclusiveAdjWithPCode

This method accepts transaction data and performs tax inclusive adjustments in order to arrive at the base sale amount and taxes for the desired total charge. Ship from and ship to information is passed using PCodes.

*Return Type:*

[TaxInclusiveTaxResults](#)

Contains the calculated base sale amount for the transaction and an array of [TaxData](#) objects that contain the information about the taxes applied.

*Parameters:*

[SalesUseTransaction](#)

Sales & Use transaction data

[Nexus](#) []

An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion](#) []

An array of [Exclusion](#) objects. Can be null or an empty list.

*Errors:*

- ShipToPCode must be set! – The ShipToPCode property of the SalesUseTransaction was not set.
- ShipFromPCode must be set! – The ShipFromPCode property of the SalesUseTransaction was not set.
- Adjustment is null! – No Transaction passed in
- PCode not found. – One of the two PCode values passed in is not in AFC

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

The desired total charge must be placed in the Charge field of the [SalesUseTransaction](#) as a positive number.

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

Adjustment methods are used to process refunds or credits. They rely on the [AdjustmentMethod field](#).

Discounts should be processed using the [DiscountAttributes](#) object in one of the tax calculation methods, not by calling the adjustment methods.

## 8.17 SAUCalcTaxInclusiveAdjWithFipsCode

This method accepts transaction data and performs tax inclusive adjustments in order to arrive at the base sale amount and taxes for the desired total charge. Ship from and ship to information is passed using Fips Codes.

### Return Type:

[TaxInclusiveTaxResults](#)

Contains the calculated base sale amount for the transaction and an array of [TaxData](#) objects that contain the information about the taxes applied.

### Parameters:

[SalesUseTransaction](#)

Sales & Use transaction data

[Nexus](#) []

An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion](#) []

An array of [Exclusion](#) objects. Can be null or an empty list.

### Errors:

- ShipToFipsCode must be set! – The ShipToFipsCode property of the SalesUseTransaction was not set.
- ShipFromFipsCode must be set! – The ShipFromPCode property of the SalesUseTransaction was not set.
- Adjustment is null! – No Transaction passed in
- Fips Code not found. – One of the two Fips Code values passed in is not in AFC
- FIPS (to PCode) cross-reference database not open. - – Indicates an error occurred on the server

See also [Error Messages Common to all Calculation Methods](#)

### Remarks:

The desired total charge must be placed in the Charge field of the [SalesUseTransaction](#) as a positive number.

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

Adjustment methods are used to process refunds or credits. They rely on the [AdjustmentMethod field](#).

Discounts should be processed using the [DiscountAttributes](#) object in one of the tax calculation methods, not by calling the adjustment methods.

## 8.18 SAUCalcTaxInclusiveAdjWithZipAddress

This method accepts transaction data and performs tax inclusive adjustments in order to arrive at the base sale amount and taxes for the desired total charge. Ship from and ship to information is passed using [ZipAddress](#) objects.

### Return Type:

[TaxInclusiveTaxResults](#)

Contains the calculated base sale amount for the transaction and an array of [TaxData](#) objects that contain the information about the taxes applied.

### Parameters:

[SalesUseTransaction](#)

Sales & Use transaction data

[Nexus](#) []

An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion](#) []

An array of [Exclusion](#) objects. Can be null or an empty list.

### Errors:

- ShipToZipAddress must be set! – The ShipToZipAddress property of the SalesUseTransaction was not set.
- ShipFromZipAddress must be set! – The ShipFromZipAddress property of the SalesUseTransaction was not set.
- Adjustment is null! – No Adjustment passed in
- County/State/Zip not found. – One of the two ZipAddress values passed in is not in AFC

See also [Error Messages Common to all Calculation Methods](#)

### Remarks:

The desired total charge must be placed in the Charge field of the [SalesUseTransaction](#) as a positive number.

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

Adjustment methods are used to process refunds or credits. They rely on the [AdjustmentMethod field](#).

Discounts should be processed using the [DiscountAttributes](#) object in one of the tax calculation methods, not by calling the adjustment methods.

## 8.19 BeginCustomerBatch

Flags the beginning of a customer transaction batch and returns a customer batch ID that can be used to submit each transaction and process the entire batch.

### Return Type:

*Parameters:*  
None

## 8.20 SAUCalcCustTaxes

Submits a single transaction to be processed for a customer batch.

*Return Type:*

bool	Boolean indicating success or failure
------	---------------------------------------

*Parameters:*

<i>long</i>	<i>Customer batch identifier</i>
<a href="#"><u>SalesUseTransaction</u></a>	Sales and use transaction data

*Errors:*

- Ship-From location not specified. – The origination location of the Transaction was not set.
- Ship-to location not specified. – The termination location of the Transaction was not set.
- Transaction is null – No transaction passed in.
- Invalid customer batch identifier.
- Maximum number of customer transaction exceeded.

See also [Error Messages Common to all Calculation Methods](#)

Remarks:

Adjustment transactions are primarily used to process refunds or credits. They rely on the [AdjustmentMethod](#) field.

Discounts can also be processed by setting the [DiscountType](#) field in the adjustment transaction.

## 8.21 SAUCalcCustAdj

Submits a single adjustment to be processed for a customer batch.

*Return Type:*

bool	Boolean indicating success or failure
------	---------------------------------------

*Parameters:*

Long	Customer batch identifier
SalesUseTransaction	Sales and use transaction data

*Errors:*

- Ship-From location not specified. – The origination location of the Transaction was not set.

- Ship-to location not specified. – The termination location of the Transaction was not set.
- Adjustment is null – No adjustment passed in.
- Invalid customer batch identifier.
- Maximum number of customer transaction exceeded.

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

Adjustment transactions are primarily used to process refunds or credits. They rely on the [AdjustmentMethod field](#).

Discounts can also be processed by setting the [DiscountType field](#) in the adjustment transaction.

## 8.22 ProcessCustomerBatch

Processes a customer batch and returns the tax calculation results.

*Return Type:*

CustomerResults	Data structure containing an array of <a href="#">TaxData</a> objects for each individual transaction processed and <a href="#">CustomerTaxData</a> array containing the summarized taxes for the customer batch.
-----------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

*Parameters:*

long	Customer batch ID
bool	Boolean indicating if the detailed taxes for each individual line item should be returned in the results.
<a href="#">Nexus</a> []	An array of <a href="#">Nexus</a> objects. Can be null or an empty list.
<a href="#">Exclusion</a> []	An array of <a href="#">Exclusion</a> objects. Can be null or an empty list.

*Errors:*

- Invalid customer batch identifier – Customer batch ID is not valid.

See also [Error Messages Common to all Calculation Methods](#)

**NOTE:** This API does not support tax inclusive calls. Please see ProcessCustomerBatchV2 for tax inclusive transactions.



## 8.23 ProcessCustomerBatchV2

Processes a customer batch and returns the tax calculation results. Individual taxes returned use the [TaxDataV2](#) data structure which contains a TransCharge field for the calculated base sale amount for each corresponding transaction.

### Return Type:

[CustomerResultsV2](#)

Data structure containing an array of [TaxDataV2](#) objects for each individual transaction processed and [CustomerTaxData](#) array containing the summarized taxes for the customer batch.

### Parameters:

long

Customer batch ID

bool

Boolean indicating if the detailed taxes for each individual line item should be returned in the results.

[Nexus](#) []

An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion](#) []

An array of [Exclusion](#) objects. Can be null or an empty list.

### Errors:

- Invalid customer batch identifier – Customer batch ID is not valid.

See also [Error Messages Common to all Calculation Methods](#)

**NOTE:** Use of this API does incorporate use of tax inclusive transactions

## 8.24 CalcTaxesInCustMode

Processes a batch of telecom and/or sales and use transactions and adjustments in Invoice Mode.

### Return Type:

CustomerResults

Data structure containing an array of [TaxData](#) objects for each individual transaction processed and [CustomerTaxData](#) array containing the summarized taxes for the customer batch.

### Parameters:

[Transaction](#) []

Telecom transaction array (See **TM\_00116\_AFC SaaS Pro Telecom Developer Manual**).

[Transaction](#) []

Telecom transaction array to be processed as adjustments (See **TM\_00116\_AFC SaaS Pro Telecom Developer Manual**).

[SalesUseTransaction](#) []

Sales and use transaction array.

[SalesUseTransaction](#) []

Sales and use transaction array to be processed as adjustments.

[Nexus](#) []

An array of [Nexus](#) objects. Can be null or an empty list.

Nexus only applies Sales and Use transactions.

<a href="#">Exclusion</a> []	An array of <a href="#">Exclusion</a> objects. Can be null or an empty list.
bool	Boolean indicating if the detailed taxes for each individual line item should be returned in the results.

*Errors:*

See [Error Messages Common to all Calculation Methods](#)

**NOTE:** If a transaction in the batch generates an error, the entire batch will fail. Also, this API does not support tax inclusive calls. Please see ProcessCustomerBatchV2 for tax inclusive transactions.

## 8.25 CalcTaxesInCustModeV2

Processes a batch of telecom and/or sales and use transactions and adjustments in Invoice Mode. Individual taxes returned use the [TaxDataV2](#) data structure which contains a TransCharge field for the calculated base sale amount for each corresponding transaction.

*Return Type:*

<a href="#">CustomerResultsV2</a>	Data structure containing an array of <a href="#">TaxDataV2</a> objects for each individual transaction processed and <a href="#">CustomerTaxData</a> array containing the summarized taxes for the customer batch.
-----------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

*Parameters:*

Transaction[]	Telecom transaction array (See <b>TM_00116_AFC SaaS Pro Telecom Developer Manual</b> ).
Transaction[]	Telecom transaction array to be processed as adjustments (See <b>TM_00116_AFC SaaS Pro Telecom Developer Manual</b> ).
<a href="#">SalesUseTransaction</a> []	Sales and use transaction array.
<a href="#">SalesUseTransaction</a> []	Sales and use transaction array to be processed as adjustments.
<a href="#">Nexus</a> []	An array of <a href="#">Nexus</a> objects. Can be null or an empty list. Nexus only applies Sales and Use transactions.
<a href="#">Exclusion</a> []	An array of <a href="#">Exclusion</a> objects. Can be null or an empty list.
bool	Boolean indicating if the detailed taxes for each individual line item should be returned in the results.

*Errors:*

See [Error Messages Common to all Calculation Methods](#)

**NOTE:** If a transaction in the batch generates an error, the entire batch will fail. Also, use of this API does incorporate use of tax inclusive transactions.

## 8.26 SAUCalcJurisdiction

Determines the taxing jurisdiction for a transaction and returns the PCode at the lowest jurisdiction level.

### Return Type:

uint PCode for taxing jurisdiction.

### Parameters:

[SalesUseTransaction](#) Sales and use transaction data

### Errors:

- Ship-From location not specified. – The origination location of the Transaction was not set.
- Ship-to location not specified. – The termination location of the Transaction was not set.
- Transaction is null! – No Transaction passed in
- PCode not found. – One of the two PCode values passed in is not in AFC
- FIPS not found. – One of the two FipsCode values passed in is not in AFC
- County/State/Zip not found. – One of the two ZipAddress values passed in is not in AFC

See also [Error Messages Common to all Calculation Methods](#)

### Remarks:

None.

NOTE: The transaction/service pair is required and must be provided in order to determine the correct tax jurisdiction.

Also, please note if SAUCalcJurisdiction is called along with an Exclusion, the Exclusion will not be applied.

## 8.27 SAUCalcTaxesWithOverrides

Accepts transaction data and tax rate override data to perform tax calculations with provided override information. Ship from and Ship to information may be entered as PCode, FIPS Codes, [ZipAddress](#) or NPANXX.

### Return Type:

[TaxData](#)[] An array of [TaxData](#) objects that contain the information about the taxes applied.

### Parameters:

[SalesUseTransaction](#) Sales and use transaction data  
[TaxRateOverrideInfo](#)[] Tax rate override data

### Errors:

- The ship-from jurisdiction has not been specified. – Indicates an error occurred on the server

- The ship-to jurisdiction has not been specified. – Indicates an error occurred on the server
- PCode not found. – One of the three PCode values passed in is not in AFC
- Transaction is null! – No Transaction passed in
- FIPS not found. – One of the three FipsCode values passed in is not in AFC
- County/State/Zip not found. – One of the three ZipAddress values passed in is not in AFC
- NPANXX not found. – One of the three NpaNxx values passed in is not in AFC
- This method cannot be used if you have override file in place. – Override file is in place.
- AFC failed to insert the override. – Override data cannot be applied.

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

This method cannot be used if the user already has an override file in place.

## 8.28 SAUCalcAdjWithOverrides

Accepts transaction data and tax rate override data to perform tax adjustment calculations with provided override information. Ship from and Ship to information may be entered as PCode, FIPS Codes, [ZipAddress](#) or NPANXX.

*Return Type:*

[TaxData\[\]](#) An array of [TaxData](#) objects that contain the information about the taxes applied.

*Parameters:*

[SalesUseTransaction](#) Sales and use transaction data  
[TaxRateOverrideInfo\[\]](#) Tax rate override data

*Errors:*

- The ship-from jurisdiction has not been specified. – Indicates an error occurred on the server
- The ship-to jurisdiction has not been specified. – Indicates an error occurred on the server
- PCode not found. – One of the three PCode values passed in is not in AFC
- Transaction is null! – No Transaction passed in
- FIPS not found. – One of the three FipsCode values passed in is not in AFC
- County/State/Zip not found. – One of the three ZipAddress values passed in is not in AFC
- NPANXX not found. – One of the three NpaNxx values passed in is not in AFC
- This method cannot be used if you have override file in place. – Override file is in place.
- AFC failed to insert the override. – Override data cannot be applied.

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

This method cannot be used if the user already has an override file in place.

Adjustments require application of the same override data that is used with SAUCalcTaxesWithOverrides.

Adjustment transactions are primarily used to process refunds or credits. They rely on the [AdjustmentMethod field](#).

Discounts can also be processed by setting the [DiscountType field](#) in the adjustment transaction.

## 8.29 CommitTransactions

The CommitTransactions API is used to commit or uncommit a DocumentCode.

### *Return Type:*

bool                                      Boolean indicating success or failure

**Note:** Exception thrown with appropriate error message in case of failure.

### *Parameters:*

[SalesUseTransaction](#)  
[CommitData](#)

Sales and use transaction data

Required as a parameter in order to specify a valid DocumentCode and a Boolean indicating whether transactions with the specified DocumentCode should be committed or uncommitted. Any optional field values provided in this parameter are used when reports are generated in order to replace the corresponding values originally specified within the transaction at the time of tax calculation processing.

### *Errors:*

- DocumentCode cannot be blank or null.
- DocumentCode not found.
- DocumentCode has been locked.
- DocumentCode cannot exceed 150 characters.

See also [Error Messages Common to all Calculation Methods](#)

### *Remarks:*

None.

## 8.30 Error Messages Common to all Calculation Methods

These messages can be returned by any of the calculation methods:

- Sale Attributes is Required! – No Sale Attributes were passed in
- AdjustmentMethod is Invalid! – The Adjustment Method passed in is not valid
- FOB is Invalid! – The FOB value passed in is not valid
- No Session Available! – Indicates that no sessions were available for your request. You may be making too many simultaneous requests to the web service or there is an issue that needs to be reported.
- Authorization Error – Indicates that your userid or password is not recognized.
- Not Authorized! – Indicates that you have called a method on the web service that your service agreement does not include.
- Server Fault! – Indicates an error occurred on the server.
- CustomerType is Invalid! – The Customer Type passed in is not valid
- BusinessClass is Invalid! – The Business Class passed in is not valid
- ServiceClass is Invalid! – The Service Class passed in is not valid
- Jurisdiction not found. – The jurisdiction does not exist in AFC
- Invalid Transaction Date – The Transaction Date passed in is not valid
- Invalid transaction/service pair – The Transaction Type and Service Type combination is not valid
- Transaction/Service pair is not supported by the API call – The Transaction Type and Service Type combination cannot be used by the Sales & Use Interface
- Invalid sales attribute – The sales attribute type passed in is not valid
- Sales attribute contains an invalid property value – One of the property values passed in is not valid
- Session not initialized – Indicates an error occurred on the server
- JCode database not open. – Indicates an error occurred on the server
- PCode database not open. – Indicates an error occurred on the server
- ZipCode database not open. – Indicates an error occurred on the server
- NPANXX database not open. – Indicates an error occurred on the server
- Address database not open. – Indicates an error occurred on the server
- Unable to start AFC. – Indicates an error occurred on the server
- filelocs.txt configuration file not found. – Indicates an error occurred on the server
- Unable to read filelocs.txt configuration file – Indicates an error occurred on the server
- An EZtaxSession hasn't been specified for this transaction – Indicates an error occurred on the server
- The EZtaxSession object has been disposed. – Indicates an error occurred on the server
- Unable to return the log information – Indicates an error occurred on the server
- The ship-from jurisdiction has not been specified. – Indicates an error occurred on the server
- The ship-to jurisdiction has not been specified. – Indicates an error occurred on the server
- PCode not found.
- Address not found.

## 9. Sales and Use Invoice Mode Web Methods

**Note:** The APIs identified and described in this section of the document should no longer be used and will be deprecated in the future. Please refer to and use the APIs in [Section 7 Sales and Use Web Methods](#).

These methods process each an array of transactions for the same customer. See Section 6 for the Non Invoice Mode web methods.

Any of these messages can throw an exception. Errors can occur in any of the calculation methods are listed in [Error Messages Common to all Calculation Methods](#). Messages that are specific to a particular method are shown with each method description.

Method Name	Summary
<a href="#">SAUCalcTaxesAndAdjWithPCodeInCustMode</a>	Calculate taxes and adjustments on supplied arrays of transactions using PCodes
<a href="#">SAUCalcTaxesAndAdjWithFipsCodeInCustMode</a>	Calculate taxes and adjustments on supplied arrays of transactions using FIPS Codes
<a href="#">SAUCalcTaxesAndAdjWithZipAddressInCustMode</a>	Calculate taxes and adjustments on supplied arrays of transactions using Zip Addresses

### 9.1 SAUCalcTaxesAndAdjWithPCodeInCustMode

This method accepts an array of transaction data and performs appropriate tax calculations and adjustments. Ship from and ship to information is passed using PCodes.

*Return Type:*

[CustomerTaxData](#)[] An array of [CustomerTaxData](#) objects that contain the information about the taxes applied.

*Parameters:*

[SalesUseTransaction](#)[] An array of Sales & Use transaction data for tax calculations  
[SalesUseTransaction](#)[] An array of Sales & Use transaction data for adjustments  
[Nexus](#) [] An array of [Nexus](#) objects. Can be null or an empty list.  
[Exclusion](#) [] An array of [Exclusion](#) objects. Can be null or an empty list.

*Errors:*

- ShipToPCode must be set! – The ShipToPCode property of the SalesUseTransaction was not set.
- ShipFromPCode must be set! – The ShipFromPCode property of the SalesUseTransaction was not set.
- Transaction is null! – A null Transaction object was passed in
- Adjustment is null! – A null Adjustment object was passed in
- PCode not found. – One of the two PCode values passed in is not in AFC

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

Adjustment transactions are used to process refunds or credits. They rely on the [AdjustmentMethod](#) field.

Discounts should be processed using the [DiscountAttributes](#) object in one of the tax calculation transactions, not by using an adjustment transaction.

## 9.2 SAUCalcTaxesAndAdjWithFipsCodeInCustMode

This method accepts an array of transaction data and performs appropriate tax calculations and adjustments. Ship from and ship to information is passed using FIPS Codes.

### Return Type:

[CustomerTaxData](#)[] An array of [CustomerTaxData](#) objects that contain the information about the taxes applied.

### Parameters:

<a href="#">SalesUseTransaction</a> []	An array of Sales & Use transaction data for tax calculations
<a href="#">SalesUseTransaction</a> []	An array of Sales & Use transaction data for adjustments
<a href="#">Nexus</a> []	An array of <a href="#">Nexus</a> objects. Can be null or an empty list.
<a href="#">Exclusion</a> []	An array of <a href="#">Exclusion</a> objects. Can be null or an empty list.

### Errors:

- ShipToFipsCode must be set! – The ShipToFipsCode property of the SalesUseTransaction was not set.
- ShipFromFipsCode must be set! – The ShipFromFipsCode property of the SalesUseTransaction was not set.
- Transaction is null! – A null Transaction object was passed in
- Adjustment is null! – A null Adjustment object was passed in
- FIPS not found. – One of the two FipsCode values passed in is not in AFC SaaS Pro
- FIPS (to PCode) cross-reference database not open. - – Indicates an error occurred on the server

See also [Error Messages Common to all Calculation Methods](#)

### Remarks:

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

Adjustment transactions are used to process refunds or credits. They rely on the [AdjustmentMethod](#) field.

Discounts should be processed using the [DiscountAttributes](#) object in one of the tax calculation transactions, not by using an adjustment transaction.



### 9.3 SAUCalcTaxesAndAdjWithZipAddressInCustMode

This method accepts an array of transaction data and performs appropriate tax calculations and adjustments. Ship from and ship to information is passed using [ZipAddress](#) objects.

*Return Type:*

[CustomerTaxData](#)[]                      An array of [CustomerTaxData](#) objects that contain the information about the taxes applied.

*Parameters:*

[SalesUseTransaction](#)[]                      An array of Sales & Use transaction data for tax calculations

[SalesUseTransaction](#)[]                      An array of Sales & Use transaction data for adjustments

[Nexus](#) []                                          An array of [Nexus](#) objects. Can be null or an empty list.

[Exclusion](#) []                                      An array of [Exclusion](#) objects. Can be null or an empty list.

*Errors:*

- ShipToZipAddress must be set! – The ShipToZipAddress property of the SalesUseTransaction was not set.
- ShipFromZipAddress must be set! – The ShipFromZipAddress property of the SalesUseTransaction was not set.
- Transaction is null! – A null Transaction object was passed in
- Adjustment is null! – A null Adjustment object was passed in
- County/State/Zip not found. – One of the two ZipAddress values passed in is not in AFC SaaS Pro

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

This currently works only for the U.S. jurisdictions.

The AFC technique for determining [Nexus](#) and [Exclusions](#) are discussed in Chapter 1.

Adjustment transactions are used to process refunds or credits. They rely on the [AdjustmentMethod](#) field.

Discounts should be processed using the [DiscountAttributes](#) object in one of the tax calculation transactions, not by using an adjustment transaction.

## 10. Utility Web Methods

These methods provide a variety of information from the AFC SaaS Pro Web Service.

Method Name	Summary
<a href="#">GetAddress</a>	Returns an array of addresses that correspond to a PCode
<a href="#">GetTaxCategory</a>	Returns the tax category for a tax type
<a href="#">GetTaxDescription</a>	Returns the tax description for a tax type
<a href="#">GetTaxRates</a>	Returns the tax rate information for a jurisdiction
<a href="#">FipsToPCode</a>	Returns the PCode that corresponds to a FIPS code
<a href="#">PCodeToFips</a>	Returns the FIPS code that corresponds to a PCode
<a href="#">ZipToPCode</a>	Returns the PCode that corresponds to a Zip Address
<a href="#">NpaNxxToPCode</a>	Returns the PCode that corresponds to an NpaNxx value
<a href="#">GetServerTime</a>	Returns the server time on the AFC SaaS Pro Web Service
<a href="#">GetVersion</a>	Returns the version of the AFC SaaS Pro Web Service
<a href="#">GetEZtaxVersion</a>	Returns the version of the underlying AFC engine
<a href="#">GetEZtaxDbVersion</a>	Returns the version of the database used by the underlying AFC engine
<a href="#">CreateReport</a>	Submits a request for generating a report based on transactions and taxes that have been processed in the web service.
<a href="#">GetOptionalFieldKeyDesc</a>	Returns the description for each optional field key in the <a href="#">OptionalField</a> array of the <a href="#">SalesUseTransaction</a> . The values specified will be used when generating reports that contain these optional fields.
<a href="#">UpdateOptionalFieldKeyDesc</a>	Updates the description for each optional field key thereby indicating what each field in the <a href="#">OptionalField</a> array of the <a href="#">SalesUseTransaction</a> is used for (e.g. invoice number, line item number, etc.). The values specified will be used when generating reports that contain these optional fields.
<a href="#">ZipLookup</a>	Returns all or multiple PCodes and jurisdiction details associated with the address input provided by clients.

### 10.1 GetAddress

This method returns the addresses for the specified jurisdiction. If the jurisdiction is invalid the return will be NULL.

*Return Type:*

[AddressData](#)[]

An array of [AddressData](#) objects that contain the addresses for the jurisdiction specified by the supplied PCode.

*Parameters:*

uint

The PCode for the desired jurisdiction.

*Errors:*

- No Session Available! – Indicates that no sessions were available for your request. You may be making too many simultaneous requests to the web service or there is an issue that needs to be reported.

- Remarks:*  
Some jurisdictions have multiple valid addresses.

This method returns the tax category for the specified tax type. If the tax type is invalid the return will be NULL or an empty string.

*Return Type:*

string	A string describing the tax category for the specified tax type.
--------	------------------------------------------------------------------

*Parameters:*

int	The tax type.
-----	---------------

*Errors:*

- No Session Available! – Indicates that no sessions were available for your request. You may be making too many simultaneous requests to the web service or there is an issue that needs to be reported.
- Authorization Error – Indicates that your userid or password is not recognized.
- Not Authorized! – Indicates that you have called a method on the web service that your service agreement does not include.
- Server Fault! – Indicates an error occurred on the server.

Remarks:

None.

This method returns the description for the specified tax type. If the tax type is invalid the return will be NULL or an empty string.

*Return Type:*

string	A string describing the specified tax type.
--------	---------------------------------------------

Parameters:

int	The tax type.
-----	---------------

*Errors:*

- No Session Available! – Indicates that no sessions were available for your request. You may be making too many simultaneous requests to the web service or there is an issue that needs to be reported.
- Authorization Error – Indicates that your userid or password is not recognized.
- Not Authorized! – Indicates that you have called a method on the web service that your service agreement does not include.
- Server Fault! – Indicates an error occurred on the server.

*Remarks:*

None.

## 10.4 GetTaxRates

Determines the tax rate information for a jurisdiction identified by the input PCode.

*Return Type:*

[TaxRateInfo](#)

Tax Rate information for a jurisdiction.

*Parameters:*

uint

PCode for taxing jurisdiction

*Errors:*

- PCode not found. – The PCode value passed in is not in AFC SaaS Pro.

See also [Error Messages Common to all Calculation Methods](#)

*Remarks:*

None.

## 10.5 FipsToPCode

This method returns the PCode for the specified FIPS code. If the FIPS Code is invalid or has no AFC jurisdiction the return will be NULL.

*Return Type:*

uint

A nullable unsigned int for the PCode.

*Parameters:*

string

The FIPS code.

*Errors:*

- No Session Available! – Indicates that no sessions were available for your request. You may be making too many simultaneous requests to the web service or there is an issue that needs to be reported.
- Authorization Error – Indicates that your userid or password is not recognized.
- Not Authorized! – Indicates that you have called a method on the web service that your service agreement does not include.

## Remarks:

None.

## 10.6 PCodeToFips

This method returns the FIPS Code for the specified PCode. If the PCode Code is invalid the return will be NULL.

*Return Type:*

string	A string for the FIPS Code.
--------	-----------------------------

*Parameters:*

uint The PCode.

Errors:

- No Session Available! – Indicates that no sessions were available for your request. You may be making too many simultaneous requests to the web service or there is an issue that needs to be reported.
- Authorization Error – Indicates that your userid or password is not recognized.
- Not Authorized! – Indicates that you have called a method on the web service that your service agreement does not include.
- Server Fault! – Indicates an error occurred on the server.

## Remarks:

None.

## 10.7 ZipToPCode

This method returns the PCode for the specified ZIP Address. If the ZIP address is invalid or has no AFC jurisdiction the return will be NULL.

*Return Type:*

uint	A nullable unsigned int for the PCode. Null indicates an invalid address.
------	---------------------------------------------------------------------------

*Parameters:*

[ZipAddress](#) The [ZipAddress](#) object.

*Errors:*

- No Session Available! – Indicates that no sessions were available for your request. You may be making too many simultaneous requests to the web service or there is an issue that needs to be reported.
- Authorization Error – Indicates that your userid or password is not recognized.
- Not Authorized! – Indicates that you have called a method on the web service that your service agreement does not include.
- Server Fault! – Indicates an error occurred on the server.

*Remarks:*

Based upon the best match, only one PCode is returned. To obtain a full list of matching PCodes, please reference the [ZipLookup](#) API. Also, please note it is not required to provide the State. Please reference [Zip Lookup Requests](#) for additional information and details.

## 10.8 NpaNxxToPCode

This method returns the PCode for the specified NpaNxx value. If the NpaNxx value is invalid or has no AFC jurisdiction the return will be NULL.

*Return Type:*

uint A nullable unsigned int for the PCode

*Parameters:*

uint The NpaNxx value.

*Errors:*

- No Session Available! – Indicates that no sessions were available for your request. You may be making too many simultaneous requests to the web service or there is an issue that needs to be reported.
- Authorization Error – Indicates that your userid or password is not recognized.
- Not Authorized! – Indicates that you have called a method on the web service that your service agreement does not include.
- Server Fault! – Indicates an error occurred on the server.
- PCode not found.

*Remarks:*

None.

## 10.9 GetServerTime

This method returns the time on the AFC SaaS Pro Web Service.

*Return Type:*

DateTime                                      The AFC SaaS Pro Web Service server time.

*Parameters:*

None

*Errors:*

- No Session Available! – Indicates that no sessions were available for your request. You may be making too many simultaneous requests to the web service or there is an issue that needs to be reported.
- Authorization Error – Indicates that your userid or password is not recognized.
- Not Authorized! – Indicates that you have called a method on the web service that your service agreement does not include.
- Server Fault! – Indicates an error occurred on the server.

*Remarks:*

None.

## 10.10 GetVersion

This method returns the time on the AFC SaaS Pro Web Service.

*Return Type:*

string                                        The version of the AFC SaaS Pro Web Service.

*Parameters:*

None

*Errors:*

- Authorization Error – Indicates that your userid or password is not recognized.
- Server Fault! – Indicates an error occurred on the server.

*Remarks:*

None.





*Parameters:*

[ReportOptions](#)

Data structure containing the options for the report.

*Errors:*

- reportOptions parameter must be specified.
- ReportType must be specified.
- StartDate for the report may not be a future date.
- StartDate for the report may not be prior than 91 days.
- TimeSpan cannot be greater than 31 days.
- TimeSpan or EndDate must be specified.
- EndDate cannot be more than 31 days apart from StartDate.
- EndDate must be greater than StartDate.
- Precision must be between 0 and 6.
- FileName may not exceed 100 characters
- FileExtension may not exceed 5 characters.
- EmailAddress cannot be longer than 100 characters.
- EmailAddress is not in a valid format.
- CustomLogFields must be specified if BaseReport is not used.

*Remarks:*

None.

**Note:** Only transactions and taxes that include the company's 3-character company code in the CompanyIdentifier field of the transaction will be included in the report.

## 10.14 GetOptionalFieldKeyDesc

Returns the description for each optional field key in the [OptionalField](#) array of the [SalesUseTransaction](#). The values specified will be used when generating reports that contain these optional fields.

*Return Type:*

[OptionalKey\[\]](#)

Array containing key numbers and descriptions for each key. The OptionalKeyDesc field of the OptionalKey object contains the description for the OptionalKeyNo.

*Parameters:*

None

*Errors:*

- Authorization Error – Indicates that your userid or password is not recognized.
- Server Fault! – Indicates an error occurred on the server.

*Remarks:*

None.

## 10.15 UpdateOptionalFieldKeyDesc

Updates the description for each optional field key thereby indicating what each field in the [OptionalField](#) array of the [SalesUseTransaction](#) is used for (e.g. invoice number, line item number, etc.). The values specified will be used when generating reports that contain these optional fields.

### Return Type:

Boolean Indicates success or failure.

### Parameters:

[OptionalKey\[\]](#) Array containing key numbers and descriptions for each key. The OptionalKeyDesc field of the OptionalKey object contains the description for the OptionalKeyNo.

### Errors:

- Authorization Error – Indicates that your userid or password is not recognized.
- Server Fault! – Indicates an error occurred on the server.

### Remarks:

None.

## 10.16 ZipLookup

Returns all or multiple PCodes and jurisdiction details associated with the address input provided by clients.

### Return Type:

[ZipLookupResult](#) The [ZipLookupResult](#) object.

### Parameters:

[ZipLookup](#) The [ZipLookup](#) object.

### Errors:

- No Session Available! – Indicates that no sessions were available for your request. You may be making too many simultaneous requests to the web service or there is an issue that needs to be reported.
- Authorization Error – Indicates that your userid or password is not recognized.
- Not Authorized! – Indicates that you have called a method on the web service that your service agreement does not include.
- Server Fault! – Indicates an error occurred on the server.
- No jurisdiction data set – Indicates the request had no valid jurisdiction data set (at a minimum, one jurisdiction field must be set).
- Address not found.

### Remarks:

If Country is not specified, the default value is set to “USA”.

## 11. Web Service Data Definitions

SalesUseTransaction			
Data Definition			
	Property Name	Type	Description
	CustomerType	int	One of the <a href="#">CustomerType</a> enums
	<del>Sale</del>	bool	This field is deprecated. Use SaleOrConsumed field instead. true = Sale false = Resale (Wholesale)
	SaleOrConsumed	int	One of the <a href="#">SaleOrConsumed</a> enums. If null, Sale field will be used instead.
	TransactionType	short	See the Transaction Mapping guidelines for valid transaction/service pairs
	ServiceType	short	
	SaleAttributes	object	Must be one of the following types of objects: <a href="#">DefaultAttributes</a> <a href="#">DemurrageAttributes</a> <a href="#">TradeInAttributes</a> <a href="#">DepositAttributes</a> <a href="#">FreightAttributes</a> <a href="#">DiscountAttributes</a> <a href="#">FinanceAttributes</a> <a href="#">InstallationAttributes</a> <a href="#">ShipAndHandlingAttributes</a> <a href="#">SoftwareMaintAttributes</a> <a href="#">ServiceContractAttributes</a> <a href="#">MaintAgreementAttributes</a> <a href="#">FactoryWarrantyAttributes</a> <a href="#">ExtWarrantyAttributes</a>
	Date	DateTime	Transaction date
	Charge	double	Amount to tax
	Incorporated	bool	true = address inside city limits false = unincorporated address
	FederalExempt	bool	true = exempt federal level taxes
	StateExempt	bool	true = exempt state level taxes
	CountyExempt	bool	true = exempt county level taxes
	LocalExempt	bool	true = exempt local/city taxes
	FederalPCode	uint	federal PCode to exempt
	StatePCode	uint	state PCode to exempt
	CountyPCode	uint	county PCode to exempt
	LocalPCode	uint	local PCode to exempt
	Exclusions	<a href="#">Exclusion</a> []	Array of <a href="#">Exclusion</a> objects to apply to this transaction

SalesUseTransaction			
Data Definition			
	Property Name	Type	Description
	Exemptions	<a href="#">TaxExemption</a> []	Array of <a href="#">TaxExemption</a> objects to apply to this transaction
	CategoryExemptions	<a href="#">CategoryExemption</a> []	Array of <a href="#">CategoryExemption</a> objects to apply this transaction
	ExemptionType	int	set to 0 - reserved for future use
	invoiceNumber	uint	optional invoice or order number
	Optional	uint	optional field
	CustomerNumber	string	optional customer account number (up to 20 characters)
	CompanyIdentifier	string	optional company identifier (up to 20 characters)
	OptionalAlpha1	string	optional field up to 20 characters
	Optional4	uint	optional field
	Optional5	uint	optional field
	Optional6	uint	optional field
	Optional7	uint	optional field
	Optional8	uint	optional field
	Optional9	uint	optional field
	Optional10	uint	For live transactions, this field must indicate the billing cycle in YYYYMM format.
	AdjustmentMethod	int	for adjustments only (otherwise zero) One of the <a href="#">AdjustmentMethod</a> enums
	FOB	Int	One of the <a href="#">FOB</a> enums
	ShipFromAddress	<a href="#">ZipAddress</a>	<a href="#">ZipAddress</a> object for ship from – required if using ZIP address calculations otherwise NULL
	ShipFromFipsCode	string	FIPS Code for ship from – required if FIPS calculation, otherwise can be NULL
	ShipFromPCode	uint?	PCode for ship from - required if PCode calculation, otherwise can be NULL
	ShipToAddress	<a href="#">ZipAddress</a>	<a href="#">ZipAddress</a> object for ship to – required if using ZIP address calculations otherwise NULL
	ShipToFipsCode	string	FIPS Code for ship to – required if FIPS calculation, otherwise can be NULL
	ShipToPCode	uint?	PCode for ship to – required if PCode calculation, otherwise can be NULL
	OptionalFields	<a href="#">OptionalField</a> []	Array of optional alphanumeric values to associate with the transaction. Each entry in the array must have an appropriate key number from 1 to 10 indicating to which optional field key the value corresponds to.
	TaxInclusive	bool	For use with <a href="#">Invoice Mode</a> APIs only. Indicates if the charge amount of the transaction includes the tax amount.

ZipAddress			
Data Definition			
	Property Name	Type	Description
	CountryISO	string	3 character ISO abbreviation

ZipAddress			
Data Definition			
	Property Name	Type	Description
	County	string	County name
	Locality	string	City name
	State	string	State Abbreviation
	ZipCode	string	ZIP Code (5 digit)
	ZipP4	string	ZIP Code extension (4 digit)

AddressData			
Data Definition			
	Property Name	Type	Description
	CountryISO	string	3 character ISO abbreviation
	County	string	County name
	Locality	string	City name
	State	string	State Abbreviation
	TaxLevel	int	one of the <a href="#">TaxLevel</a> enums
	ZipBegin	string	First 5-digit ZIP Code in range
	ZipEnd	string	Last 5-digit ZIP Code in range
	ZipP4Begin	string	First 4-digit ZIP Code extension in range
	ZipP4End	string	Last 4-digit ZIP Code extension in range

Nexus			
Data Definition			
	Property Name	Type	Description
	CountryISO	string	3 character ISO abbreviation
	HasNexus	bool	true = has nexus in this state
	Locality	string	State Abbreviation

Exclusion			
Data Definition			
	Property Name	Type	Description
	CountryISO	string	3 character ISO abbreviation
	ExclusionOn	bool	true = exclude specified Country/State
	State	string	State Abbreviation

TaxExemption			
Data Definition			
	Property Name	Type	Description
	PCode	uint	PCode for jurisdiction to exempt
	TaxLevel	int	One of the <a href="#">TaxLevel</a> enums (less than 5)
	TaxType	short	Valid AFC tax type

TaxData			
Data Definition			
	Property Name	Type	Description
	AdjustmentType	int	0 if not an adjustment, otherwise one of the <a href="#">SalesDiscountType</a> enums based on user input
	Billable	bool	true = billable
	CalculationType	Int	one of the <a href="#">CalculationType</a> enums
	CategoryDescription	string	tax category description
	CategoryID	short	tax category id
	CompanyIdentifier	string	matches user input in transaction data
	Compliance	bool	true = reportable to jurisdiction
	CustomerNumber	string	matches user input in transaction data
	Description	string	tax description
	ExemptionType	int	set to 0 - reserved for future use
	ExemptSaleAmount	double	amount of sale not subject to tax
	InvoiceNumber	uint	matches user input in transaction data
	Lines	int	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.
	Locations	int	number of locations in transaction data
	Minutes	int	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.
	Optional	uint	matches user input in transaction data
	Optional4	uint	matches user input in transaction data
	Optional5	uint	matches user input in transaction data
	Optional6	uint	matches user input in transaction data
	Optional7	uint	matches user input in transaction data
	Optional8	uint	matches user input in transaction data
	Optional9	uint	matches user input in transaction data
	Optional10	uint	matches user input in transaction data
	OptionalAlpha1	string	matches user input in transaction data
	PCode	uint	PCode for the reporting jurisdiction
	Rate	double	Tax rate
	RefundUncollect	double	amount of sale refunded or written off
	ServiceLevelNumber	uint	matches user input in transaction data
	Surcharge	bool	true = tax considered FET taxable
	TaxableMeasure	double	amount of sale subject to tax
	TaxAmount	double	tax amount
	TaxLevel	int	one of the <a href="#">TaxLevel</a> enums
	TaxType	short	one of the Avalara tax types

TaxDataV2			
Data Definition			
	Property Name	Type	Description
	AdjustmentType	int	0 if not an adjustment, otherwise one of the <a href="#">DiscountType</a> enums based on user input
	Billable	bool	true = billable
	CalculationType	Int	one of the <a href="#">CalculationType</a> enums
	CategoryDescription	string	tax category description
	CategoryID	short	tax category id
	CompanyIdentifier	string	matches user input in transaction data
	Compliance	bool	true = reportable to jurisdiction
	CustomerNumber	string	matches user input in transaction data
	Description	string	tax description
	ExemptionType	int	set to 0 - reserved for future use
	ExemptSaleAmount	double	amount of sale not subject to tax
	InvoiceNumber	uint	matches user input in transaction data
	Lines	int	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.
	Locations	int	number of locations in transaction data
	Minutes	int	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.
	Optional	uint	matches user input in transaction data
	Optional4	uint	matches user input in transaction data
	Optional5	uint	matches user input in transaction data
	Optional6	uint	matches user input in transaction data
	Optional7	uint	matches user input in transaction data
	Optional8	uint	matches user input in transaction data
	Optional9	uint	matches user input in transaction data
	Optional10	uint	matches user input in transaction data
	OptionalAlpha1	string	matches user input in transaction data
	PCode	uint	PCode for the reporting jurisdiction
	Rate	double	Tax rate
	RefundUncollect	double	amount of sale refunded or written off
	ServiceLevelNumber	uint	matches user input in transaction data
	Surcharge	bool	true = tax considered FET taxable
	TaxableMeasure	double	amount of sale subject to tax
	TaxAmount	double	tax amount
	TaxLevel	int	one of the <a href="#">TaxLevel</a> enums
	TaxType	short	one of the Avalara tax types
	TransCharge	double	For tax inclusive transactions, this is the base sale amount calculated by AFC SaaS Pro required in order to arrive at the total desired charge.

CustomerResults			
Data Definition			
	Property Name	Type	Description
	Taxes	<a href="#">TaxData</a> []	Data structure containing an array of <a href="#">TaxData</a> objects for each individual transaction processed.
	SummarizedTaxes	<a href="#">CustomerTaxData</a> []	Data structure containing an array of <a href="#">CustomerTaxData</a> objects with summarized taxes for all transactions processed.

CustomerResultsV2			
Data Definition			
	Property Name	Type	Description
	Taxes	<a href="#">TaxDataV2</a> []	Data structure containing an array of <a href="#">TaxData</a> objects for each individual transaction processed.
	SummarizedTaxes	<a href="#">CustomerTaxData</a> []	Data structure containing an array of <a href="#">CustomerTaxData</a> objects with summarized taxes for all transactions processed.

CustomerTaxData			
Data Definition			
	Property Name	Type	Description
	PCode	uint	PCode for the reporting jurisdiction
	TaxType	int	one of the Avalara tax types
	TaxLevel	int	one of the <a href="#">TaxLevel</a> enums
	CalculationType	int	enum
	Rate	double	one of the <a href="#">CalculationType</a> enums
	TaxAmount	double	tax amount
	ExemptSaleAmount	double	amount of sale not subject to tax
	Lines	int	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.
	Locations	int	sum of locations from customer input
	Minutes	int	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.
	Description	string	tax description
	Surcharge	short	true = tax considered FET taxable
	MaxBase	double	max amount to which tax is applied
	MinBase	double	min amount to which tax is applied
	ExcessTax	double	rate for amount above max base
	TotalCharge	double	sum of charges for this customer
	CategoryID	short	tax category id
	CategoryDescription	string	tax category description



ReverseTaxResults			
Data Definition			
	Property Name	Type	Description
	BaseSaleAmount	double	Base sale amount calculated by AFC. This is the charge amount required in order to arrive at the total desired charge.
	Taxes	<a href="#">TaxData[]</a>	Taxes returned for the transaction.

TaxInclusiveTaxResults			
Data Definition			
	Property Name	Type	Description
	BaseSaleAmount	double	Base sale amount calculated by AFC. This is the charge amount required in order to arrive at the total desired charge.
	Taxes	<a href="#">TaxData[]</a>	Taxes returned for the transaction.

DefaultAttributes			
Data Definition			
	Property Name	Type	Description
This data definition contains no properties			

DemurrageAttributes			
Data Definition			
	Property Name	Type	Description
This data definition contains no properties			

TradeInAttributes			
Data Definition			
	Property Name	Type	Description
This data definition contains no properties			

DepositAttributes			
Data Definition			
	Property Name	Type	Description
This data definition contains no properties			

FreightAttributes			
Data Definition			
	Property Name	Type	Description
	PaidToSeller	bool	true = paid to seller
	CommonCarrier	bool	true = common carrier
	SellerReqShipping	bool	true = seller required shipping

DiscountAttributes			
Data Definition			
	Property Name	Type	Description
	DiscountType	int	one of the <a href="#">SalesDiscountType</a> enums

FinanceAttributes			
Data Definition			
	Property Name	Type	Description
	FinanceType	int	one of the <a href="#">FinanceType</a> enums

InstallationAttributes			
Data Definition			
	Property Name	Type	Description
	InstallationType	int	one of the <a href="#">InstallationType</a> enums

ShipAndHandlingAttributes			
Data Definition			
	Property Name	Type	Description
	SellerReqShipping	bool	true = seller required shipping

SoftwareMaintAttributes			
Data Definition			
	Property Name	Type	Description
	SellerRequired	bool	true = seller required
	AgreementType	int	one of the <a href="#">SoftwareAgreementType</a> enums
	UpdateType	int	one of the <a href="#">UpdateType</a> enums

ServiceContractAttributes			
Data Definition			
	Property Name	Type	Description
	SellerRequired	bool	true = seller required
	AgreementType	int	one of the <a href="#">ServiceContractAgreementType</a> enums
	TimeOfSale	int	one of the <a href="#">TimeOfSale</a> enums
	ItemType	int	one of the <a href="#">ItemType</a> enums

MaintAgreementAttributes			
Data Definition			
	Property Name	Type	Description
	SellerRequired	bool	true = seller required
	AgreementType	int	one of the <a href="#">MaintAgreementAttributes</a> enums
	TimeOfSale	int	one of the <a href="#">TimeOfSale</a> enums
	ItemType	int	one of the <a href="#">ItemType</a> enums

FactoryWarrantyAttributes			
Data Definition			
	Property Name	Type	Description
	SellerRequired	bool	true = seller required
	AgreementType	int	one of the <a href="#">FactoryWarrantyType</a> enums

ExtWarrantyAttributes			
Data Definition			
	Property Name	Type	Description
	SellerRequired	bool	true = seller required
	AgreementType	int	one of the <a href="#">ExtendedWarrantyType</a> enums
	TimeOfSale	int	one of the <a href="#">TimeOfSale</a> enums

TaxRateInfo			
Data Definition			
	Property Name	Type	Description
	TaxLevel	Short	Tax level number
	TaxType	Short	Tax Type number
	RateHistory	List of <a href="#">TaxRateHistory</a>	Tax Rate History

TaxRateHistory			
Data Definition			
	Property Name	Type	Description
	EffectiveDate	DateTime	Effect date of this tax rate
	LevelExemptible	Boolean	True or false indicate if it is level exemptible
	BracketInfo	List of <a href="#">TaxBracketInfo</a>	List of tax Brackets

TaxBracketInfo			
Data Definition			
	Property Name	Type	Description
	CountyOverrideOn	Boolean	Indicate county override
	CountyOverrideTax	Double	County override tax rate
	StateOverrideOn	Boolean	Indicate State override
	StateOverrideTax	Double	State Override tax rate
	Rate	Double	Tax Rate normally 0 to less than 1
	MaxBase	Double	The cap for this tax Rate

TaxRateOverrideInfo			
Data Definition			
	Property Name	Type	Description
	PCode	uint	PCode for the reporting jurisdiction
	Scope	Short	Scope to which the override is applied; State, County or Country
	TaxLevel	Short	Tax Level number

TaxRateOverrideInfo			
Data Definition			
	Property Name	Type	Description
	TaxType	Short	Tax Type number
	LevelExemptible	Boolean	Flag which indicates whether or not the tax can be level exempted
	BracketInfo	List of <a href="#">TaxBracketInfo</a>	List of Tax Brackets

ReportOptions			
Data Definition			
	Property Name	Type	Description
	StartDate	DateTime	Beginning of date range for data to be included in the report.
	EndDate	DateTime	End of date range for data to be included in the report. This field is not required if TimeSpan is specified.
	TimeSpan	TimeSpan	TimeSpan for the date range to be included in the report (e.g. number of days after StartDate). This field is not required if EndDate is specified.
	ReportType	String	Name of report to be generated.
	IncludeHeaders	Boolean	Indicates if the columns headers should be included in the first row of the report output.
	CreateNbaFile	Boolean	Indicates if non-billable amounts should be placed in a separate file.
	CreateNcaFile	Boolean	Indicates if non-compliance amounts should be placed in a separate file.
	FileName	String	Name for output file (up to 100 characters).
	FileExtension	String	Extension for output file (up to 5 characters).
	EmailAddress	String	Email address for receiving an email notification when the report has been generated (up to 100 characters).
	Precision	Int	Number of decimal places for computed values.
	BaseReport	String	When generating a “customlog” report, indicates which report type to use as a starting template.
	CustomLogFields	List of <a href="#">CustomLogField</a>	List of columns to include in the report when using the “customlog” report type.

CustomLogField			
Data Definition			
	Property Name	Type	Description
	Column	String	Name of column. Refer to <a href="#">4.4 – Custom Log Report Columns</a> .
	Include	Boolean	Indicates if the column should be included in the output.
	Sort	Boolean	Indicates if the column should be used for sorting.

CategoryExemption			
Data Definition			
	Property Name	Type	Description
	CountryISO	String	3 character ISO abbreviation
	State	String	State Abbreviation
	TaxCategory	short	Valid tax category ID

OptionalField			
Data Definition			
	Property Name	Type	Description
	OptionalKeyNo	short	Optional field key number (values from 1 to 10)
	OptionalValue	string	Value for optional field (up to 150 characters)

OptionalKey			
Data Definition			
	Property Name	Type	Description
	OptionalKeyNo	short	Optional field key number (values from 1 to 10)
	OptionalKeyDesc	string	Description or label for optional field key.

ZipLookup			
Data Definition			
	Property Name	Type	Description
	Country	string	3 character ISO abbreviation
	County	string	County name
	City	string	City name
	State	string	State Abbreviation
	ZipCode	string	ZIP Code (5 digit)
	BestMatchFlag	Boolean	True = Best Match logic False = Exact Match

ZipLookup			
Data Definition			
	Property Name	Type	Description
	Country	string	3 character ISO abbreviation
	County	string	County name
	City	string	City name
	State	string	State Abbreviation
	ZipCode	string	ZIP Code (5 digit)
	BestMatchFlag	Boolean	True = Best Match logic False = Exact Match
	LimitResults	Integer	Limit results for match to this value

ZipLookupResult			
Data Definition			
	Property Name	Type	Description
	InputMatchType	Best/Exact	Requested match type
	MatchCount	Integer	Number of matches returned
	MatchTypeApplied	Best/Exact	Match type used
	ResultsLimit	Integer	State Abbreviation
	LocationData	Object	Array of <a href="#">LocationData</a> objects that match requested lookup

LocationData			
Data Definition			
	Property Name	Type	Description
	PCode	Long	PCode for location
	Country	string	3 character ISO abbreviation
	County	string	County name
	City	string	City name
	State	string	State Abbreviation

CommitData				
Data Definition				
	Property Name	Value	Type	Description
	DocumentCode	User-defined, 150 character limit, Alphanumeric	String	Document code for transactions to commit or uncommit.
	Committed	True False	bool	Indicates if document code should be committed or uncommitted.
	CustomerNumber	User-defined	String	Optional. 20 characters max.
	InvoiceNumber	User-defined	Nullable<int>	Optional
	Optional	User-defined	Nullable<int>	Optional
	Optional4	User-defined	Nullable<int>	Optional
	Optional5	User-defined	Nullable<int>	Optional
	Optional6	User-defined	Nullable<int>	Optional
	Optional7	User-defined	Nullable<int>	Optional
	Optional8	User-defined	Nullable<int>	Optional
	Optional9	User-defined	Nullable<int>	Optional
	Optional10	User-defined	Nullable<int>	Optional
	OptionalAlpha	User-defined	String	Optional. 20 characters max.
	OptionalFields	User-defined	OptionalField[]	Array of <a href="#">OptionalField</a> objects.

## 12. Web Service Enumeration Definitions

### 12.1 Specifying Tax Level

TaxLevel			
Data Definition			
	Name	Value	Description
	Federal	0	Federal level tax
	State	1	State level tax
	County	2	County level tax
	Local	3	Local (city) tax
	Unincorporated	4	County unincorporated tax
	Other	5	Grouped tax result
	State_County_Local	6	Grouped tax result
	County_Local	7	Grouped tax result

### 12.2 Specifying Calculation Type

CalculationType			
Data Definition			
	Name	Value	Description
	Rate	1	tax based on charge amount
	Fixed	2	fixed tax
	PerMinute	3	tax based on minutes
	PerLine	4	tax based on number of access lines
	SelfTaxingRate	5	tax that taxes itself
	PerBracket	6	fixed tax with accumulated brackets
	FixedOnTier	7	fixed tax based on final bracket

### 12.3 Specifying Sale or Consumed

SaleOrConsumed			
Data Definition			
	Name	Value	Description
	Wholesale	0	Indicates that the item was sold to a wholeseller.
	Sale	1	Indicates that the item was sold to an end user - a retail sale.
	Consumed	2	Indicates that the item was consumed directly.
	Vendor Use	3	Indicates that the item is subject to vendor use tax.

## 12.4 Specifying FOB

FOB			
Data Definition			
	Name	Value	Description
	ShippingPoint	0	A term that signifies that the freight being shipped is the responsibility of the buyer after it leaves the point where it is shipped from.
	Destination	1	A term that signifies that the freight being shipped is the responsibility of the seller until the item being shipped is received by the buyer.

## 12.5 Specifying Adjustment Method

In certain cases, when the rate of a tax will change based upon the charge in the transaction, clients may wish to make an adjustment at a different rate than would normally apply for the transaction.

AdjustmentMethod			
Data Definition			
	Name	Value	Description
	Default	0	tax brackets applied normally
	LeastFavorableRate	1	tax brackets applied to produce smallest tax refund
	MostFavorableRate	2	tax brackets applied to produce largest tax refund

## 12.6 Specifying Discount Type

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.

SalesDiscountType			
Data Definition			
	Name	Value	Description
	None	0	Discount type not applicable. This should be the value used when the listed options do not apply. Note that choosing a value other than <b>None</b> can reduce the refund if the jurisdiction has determined that taxes are not refundable for the discount type specified.
	ThirdPartyCoupon	1	Coupons that are issued by a party other than the vendor. As part of this issuance, the third party will reimburse the vendor for any discount granted to a user of the coupon.
	VendorCoupon	2	Coupons that are issued by the vendor. As part of this issuance, the vendor does not get reimbursed for any discount granted to a user of the coupon.



SalesDiscountType			
Data Definition			
	Name	Value	Description
	DiscountCard	3	Cards given to the customer by an issuing retailer. Using this card entitles the user to get discounts only open to card holders. An example of this is a grocery store card where certain food items are discounted to card holders.
	RebateFactory	4	A rebate/reduction of price where a party is reimbursed for the value of the amount being rebated.
	RebateAutomotive	5	Same as a manufacturers rebate with the exception being that it is for an automobile. Note that some states specifically treat this rebate different than a manufacturers rebate.
	RebateRetailer	6	A rebate/reduction of price where a there is no party being reimbursed for the value of the amount being rebated.
	CashDiscount	7	A non-reimbursable discount offered to reduce the price of the taxable item or service being sold.
	EarlyPayment	8	A non-reimbursable discount granted to individuals who pay before a preset date.
	QuantityDiscount	9	A non-reimbursable discount granted to parties who purchase above a certain amount of items or services.
	TermDiscount	10	A non-reimbursable discount granted to parties who meet the terms and conditions of a sales contract between the vendor and the purchaser.
	TradeDiscount	11	A non-reimbursable discount granted to those parties in certain industries or trades.
	CouponBookRedemption	12	A non-reimbursable discount granted to those parties who use a coupon from a coupon book or redeems a gift certificate. The book or the certificate was purchased from either a retailer or third party.

## 12.7 Specifying Finance Type

FinanceType			
Data Definition			
	Name	Value	Description
	Original	1	A one-time or reoccurring fee charged to the purchaser based on the extension of credit on a transaction.
	Rebilled	2	A one-time or reoccurring fee charged to the purchaser after the sales transaction has occurred, usually based on a late payment or failure to pay.
	ThirdParty	3	Financing for a transaction is obtained from a party other than the seller or purchaser.

## 12.8 Specifying Installation Type

InstallationType			
Data Definition			
	Name	Value	Description
	None	0	
	InstallationExpenses	1	Miscellaneous expenses incurred for an installation such as inbound freight charge, supplies consumed in the installation.
	PublicUtilityService	2	The installation of a service covered by the appropriate utility regulators in that state.
	ReinstallITPP	3	The process of uninstalling a Tangible Personal Property item and reinstalling it either in the same or at a different place.
	InRealProperty	4	The act of attaching personal property to real property. The real property qualifies as new construction.
	RadioAndTelevision	5	The act of installing a television set or radio to either real or tangible personal property.
	AirConditioning	6	The act of installing an air conditioning unit to real property.
	AluminumSiding	7	The act of attaching aluminum siding to real property.
	Glass	8	The act of attaching either windows or mirrors to real property.
	FloorCoveringAttach	9	The act of attaching floor covering to real property.
	FloorCoveringStickAndPad	10	The act of attaching floor covering to real property by first glueing a pad to the floor and then sticking the carpet on top of the pad.
	GeneralSstpRule	11	

## 12.9 Specifying Software Agreement Type

SoftwareAgreementType			
Data Definition			
	Name	Value	Description
	None	0	
	Updates	1	An agreement that strictly involves updates to the software sold. This would not include any agreement that covers service of any kind.
	Service	2	An agreement that strictly involves service that would include all needed services including customer support needed to service the software. This would not include any service that only includes use of a customer support or customer help phone service.
	UpdatesAndService	3	An agreement that covers both updates and services, including those that cover updates

SoftwareAgreementType			
Data Definition			
	Name	Value	Description
			along with only a customer support or customer help phone service.
	CustomerSupport	4	An agreement that only covers call to a customer support or customer help phone service.

## 12.10 Specifying Service Contract Agreement Type

ServiceContractAgreementType			
Data Definition			
	Name	Value	Description
	Parts	1	The service contract only covers parts needed to maintain or repair the property involved. It would not cover labor used to maintain or repair the property involved.
	Labor	2	The service contract only covers labor needed to maintain or repair the property involved. It would not cover parts used to maintain or repair the property involved.
	PartsAndLabor	3	The service contract covers both parts and labor needed to maintain or repair the property involved.

## 12.11 Specifying Maintenance Agreement Type

MaintenanceAgreementType			
Data Definition			
	Name	Value	Description
	Parts	1	The maintenance agreement only covers parts needed to maintain the property involved. It would not cover labor used to maintain the property involved.
	Labor	2	The maintenance agreement only covers labor needed to maintain the property involved. It would not cover parts used to maintain the property involved.
	PartsAndLabor	3	The maintenance agreement covers both parts and labor needed to maintain the property involved.
	ServiceOnly	4	The maintenance agreement covers only the service needed to maintain the property involved

## 12.12 Specifying Factory Warranty Type

FactoryWarrantyType			
Data Definition			
	Name	Value	Description
	Parts	1	The factory warranty only covers parts needed to repair the property involved. It would not cover labor used to repair the property involved.
	Labor	2	The factory warranty only covers labor needed to repair the property involved. It would not cover parts used to maintain the property involved.
	PartsAndLabor	3	The factory warranty covers both parts and labor needed to repair the property involved.

## 12.13 Specifying Extended Warranty Type

ExtendedWarrantyType			
Data Definition			
	Name	Value	Description
	Parts	1	The extended warranty only covers parts needed to repair the property involved. It would not cover labor used to repair the property involved.
	Labor	2	The extended warranty only covers labor needed to repair the property involved. It would not cover parts used to maintain the property involved.
	PartsAndLabor	3	The extended warranty covers both parts and labor needed to repair the property involved.
	ServiceOnly	4	The extended warranty covers only the Service needed to repair the property involved

## 12.14 Specifying Update Type

UpdateType			
Data Definition			
	Name	Value	Description
	None	0	
	Tangible	1	Agreement that strictly involves updates to the software sold. The updates are delivered by using a physical medium. (i.e. updates sent to a customer by CD or disk)
	Electronic	2	Agreement that strictly involves updates to the software sold. The updates are delivered by using an electronic medium (i.e. updates sent over the internet)

### 12.15 Specifying 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.

CustomerType			
Data Definition			
	Name	Value	Description
	Residential	0	When transactions are made by a customer for home use.
	Business	1	When transactions are made at a place of business.
	SeniorCitizen	2	When transactions made by a customer meeting the jurisdiction requirements to be considered a senior citizen and qualify for senior citizen tax breaks.
	Industrial	3	When transactions are made at an industrial business.

### 12.16 Specifying Time of Sale

TimeOfSale			
Data Definition			
	Name	Value	Description
	DuringInitialSale	1	
	AfterInitialSale	2	

### 12.17 Specifying Item Type

ItemType			
Data Definition			
	Name	Value	Description
	Product	0	
	Service	1	

### 12.18 Specifying Exemption Type

- Specifying the exemption type is for reporting purposes only and does not impact tax calculations.
- Spaces and capitalization do not matter in Exemption Type value.
- The **bolded** text is sufficient, additional characters are ignored.
- The optional numeric ID can be used. For example:  
Non Profit Sales = nonprofitsales = nonprofit = 6  
Fed Coupons WIC Vouchers = Fed Coupons = FEDCOUPONS = 4

ExemptionType		
Name	Value	Description
<b>None</b> (Default)	0	Exemption Type not applicable.
<b>Fed Sales</b> Supremacy	1	Sales to the Federal government exempt based on supremacy.
<b>State Local Gov</b> Sales	2	Sales to the State or Local government in the state that you operate in.
<b>Sales for Resale</b>	3	Sales that are designated for resale.
<b>Fed Coupons</b> WIC Vouchers	4	Sales at retail for which Federal government coupons or vouchers under the supplemental feeding for women, infants and children program are used as payment.
<b>Reduced Food</b> Rate	5	Tax imposed on food that is taxed at a reduced rate.
<b>Non Profit</b> Sales	6	Sales made by or to not-for-profit civic, social, service or fraternal organizations.
<b>Public School</b> Sales	7	Sales made by or to all elementary and secondary schools operated at public expense in their educational functions and activities.
<b>Religious</b> Charitable Sales	8	Sales made by or to religious and charitable organizations and institutions in their religious, charitable or educational functions and activities.
<b>Prescription Drug</b> Sales	9	Drugs which may be legally dispensed by a licensed pharmacist only upon a lawful prescription of a practitioner licensed to administer those items.
<b>Prosth</b> Ortho Devices	10	Prosthetic or orthopedic devices as defined on January 1, 1980, by the Federal Medicare program pursuant to Title XVIII of the Social Security Act of 1965.
<b>Insulin</b> Sales	11	All sales of insulin.
<b>Interstate Foreign</b> Sales	12	Sales in commerce between various states or between any foreign country (Export Sales).
<b>Ingredient</b> Component Parts	13	Sale of ingredient or component parts.

## 13.Monthly Update

The AFC SaaS Pro 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.

**Note:** Historical and current month tax rates are not impacted by the updates. Updates are reflected in the subsequent month's tax rates.