webMethods Trading Networks Built-In Services Reference

Version 10.11

October 2021

WEBMETHODS

This document applies to webMethods Trading Networks 10.11 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: TN-BIS-1011-20211015

Table of Contents

About this Guide

This guide is for developers who want to programmatically access the functions of webMethods Trading Networks. It describes the built-in services provided with Trading Networks.

Note:
The webMethods Trading Networks and webMethods for Partners components perform the same functionality. The difference between the components is that webMethods Trading Networks allows you to have as many partners in your network as you want, and webMethods for Partners allows you to have only a single partner. This manual provides documentation for both components although it only refers to webMethods Trading Networks (referred to as Trading Networks).

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies service names and locations in the format *folder.subfolder.service*, APIs, Java classes, methods, properties.
Italic	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at https://documentation.softwareag.com.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at https://empower.softwareag.com/.

You can find product information on the Software AG Empower Product Support website at https://empower.softwareag.com.

To submit feature/enhancement requests, get information about product availability, and download products, go to Products.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the Knowledge Center.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at https://empower.softwareag.com/public_directory.aspx and give us a call.

Software AG Tech Community

You can find documentation and other technical information on the Software AG Tech Community website at https://techcommunity.softwareag.com. You can:

- Access product documentation, if you have Tech Community credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

1 Core Services

Overview

Use core services (services in the wm.tn folder) to perform the basic business document exchange functions.

Summary of Elements in this Folder

The elements that are available in this folder are listed in the following table:

Element	Description
wm.tn:log	Adds the specified information as an entry into the Trading Networks activity log.
wm.tn:receive	Receives a document for Trading Networks to recognize and process. This service recognizes the type of document and submits it to Trading Networks to perform business document exchange.
wm.tn:reroute	Locates a document that you specify in the Trading Networks database and processes it again. To process the document again, Trading Networks looks up the appropriate processing rule and performs the processing actions defined in the processing rule.
wm.tn:submit	Submits a document that has already been recognized to Trading Networks for processing.

wm.tn:log

Adds the specified information as an entry into the Trading Networks activity log.

Input Parameters

entryType **String** The type of the entry. The valid values for *entryType* are:

ERROR

WARNING

MESSAGE

entryClass **String** (optional) The category (or activity class) for the entry. The value can be any string from 1-20 characters. Trading Networks uses the following activity classes:

Delivery

Envelope

General

Processing

Saving

Recognition

Validation

Verification

For a description of the activity classes that Trading Networks uses, see *webMethods Trading Networks User's Guide*.

briefMessage	**String** A brief synopsis of the entry. The value can be any string from 1-80 characters.
fullMessage	**String** (optional) A more detailed message about the reason for adding the activity log entry. The value can be any string from 1-1024 characters.
relatedDocId	**String** (optional) The internal ID of the document related to this activity log entry.
relatedPartnerId	**String** (optional) The partner ID for the partner related to this activity log entry.
B2BUser	**String** (optional) The user name of the current user when this activity log entry is created.
relatedConversationID	**String** (optional) The conversation ID that is related to this activity log entry.

Output Parameters

updateCount	**String** Whether the entry was added or failed to be added for some reason. The following values apply: ■ `1` - The service added the entry. ■ `0` - The service did not add the entry.

Usage Notes

Use this service to log events that occur in the Trading Networks system. Because of data integrity constraints in the database, if you specify a related document, that document must already be saved. If you specify the ID of an unsaved document, or an unknown partner ID, the service does not add the activity log entry.

wm.tn:receive

Receives a document that Trading Networks is to recognize and process. This service recognizes the type of document and submits it to Trading Networks to perform business document exchange.

Important:
Although Trading Networks can process documents of any supported EDI standard, it cannot properly process a mixture of TRADACOMS and non-TRADACOMS documents in a single submission. If the first inbound document is a TRADACOMS document, Trading Networks considers any subsequent non-TRADACOMS documents to be of the Unknown document type. Similarly, if the first inbound document is a non-TRADACOMS document, Trading Networks considers any subsequent TRADACOMS documents to be of the Unknown document type.

This service ensures that the sender of the document matches the current user. If you are sending documents from within processing rules or services and this identify check fails, see wm.tn.doc.xml:routeXml.

Input Parameters

node

Object (required for XML documents; not applicable for flat file documents and EDI documents) A document to process (must be an instance of com.wm.lang.xml.Document). The typical way to get an XML document into the pipeline is by posting an XML document to Integration Server.

Note:
You can add flat file documents or EDI documents in the pipeline by adding them as an Object with the name ffdata and edidata, respectively.

TN_parms

Document (optional) An IS document (IData object) that you can use to provide parameters that govern how Trading Networks recognizes and processes a document. *TN_parms* is primarily used for flat file processing.

The document gateway service adds *hints* to *TN_parms* that Trading Networks uses when performing document recognition for a flat file document. For information about using hints in a document gateway service, see *webMethods Trading Networks Administrator's Guide*.

For both XML and flat files, optionally add the following fields:

- *TN_parms*/*DoctypeID* or *TN_parms*/*DoctypeName* to identify the TN document type to use, thus bypassing document recognition and eliminating the overhead of searching for the TN document type. If you specify both variables, *DoctypeID* is used.
  - *TN_parms*/*DoctypeID* is a string that identifies the internal identifier of the TN document type. To determine the identifier use wm.tn.doctype:list. Using *DoctypeID* rather than *DoctypeName* is more stable because the internal identifier cannot be changed.

- *TN_parms/DoctypeName* is a string that identifies the name of the TN document type. Be sure to use the exact combination of upper- and lowercase letters.

- *TN_parms/processingRuleID* or *TN_parms/processingRuleName* to identify the processing rule to use, thus bypassing the processing rule lookup and eliminating the overhead of searching for a processing rule. If you specify both variables, *processingRuleID* is used.
  - *TN_parms/processingRuleID* is a string that identifies the internal identifier of the processing rule. To determine the identifier use the wm.tn.route:list service. Using *processingRuleID* rather than *processingRuleName* is more stable because the internal identifier cannot be changed.
  - *TN_parms/processingRuleName* is a string that identifies the name of the processing rule. Be sure to use the exact combination of upper- and lowercase letters.

- *TN_parms/$bypassRouting* to indicate whether Trading Networks should use a processing rule to process the document. Set the value of *$bypassRouting* to one of the following strings:
  - `true` to disable processing rule routing. Disable the processing rule routing, for example, if a business process handles the document. When processing rule routing is disabled, Trading Networks performs the pre-processing actions identified in the TN document type; however, it does not search for a processing rule, nor perform any processing rule actions.
  - `false` to enable processing rule routing. Default. When processing rule routing is enabled, Trading Networks searches for a processing rule or uses the rule identified by *TN_parms/processingRuleID* or *TN_parms/processingRuleName* and performs the actions defined in the processing rule.

Output Parameters

bizdoc	**Document** (optional) The document that Trading Networks received (i.e. the document passed in the *node* input variable) formatted as an IS document (IData object). For the structure of *bizdoc*, see wm.tn.rec:BizDocEnvelope.
sender	**Document** (optional) The profile summary for the sender of the document. For the structure of *sender*, see wm.tn.rec:ProfileSummary.
receiver	**Document** (optional) The profile summary for the receiver of the received document. For the structure of *receiver*, see wm.tn.rec:ProfileSummary.

TN_parms	**Document** (optional) An IS document (IData object) that provides *hints* that Trading Networks uses when performing document recognition for a flat file document. For information about document gateway services hints, see *webMethods Trading Networks Administrator's Guide*.
flags	**Document** (optional) Flags that specify the pre-processing actions for the document. If specified, the service uses the *persist?* flag to determine whether to save the document. The flags must be an instance of com.wm.app.tn.route.PreProcessingFlags.

Usage Notes

- This service returns after Trading Networks completes processing for the document that is, after Trading Networks executes the pre-processing and processing actions for the document. If the processing actions instruct Trading Networks to execute a service asynchronously, the asynchronously invoked service may not be complete.
- If you are invoking this service to process documents for other components that use Trading Networks, for example webMethods Module for EDI, see the documentation for that component to determine how to submit documents to Trading Networks.
- If you invoke Core Services directly, by default *none* of the output parameters appear in the pipeline. To include output parameters in the pipeline, do the following:
  - To include *all* of the service's output parameters in the pipeline (as well as the input parameter's *node* object), include the Trading Networks parameter *clearTNObjects* in the *TN_parms* parameter and set it to false as follows:

    ```
    clearTNObjects=false
    ```

  - To clear the pipeline of only *some* output parameters, specify the parameter *clearKeys* in the *TN_parms* parameter, and set the value as a comma-separated list of those parameters. For example, if the service is receiving an XML document, to clear the pipeline of *node*, *bizdoc*, *sender*, and *receiver* for this service, specify:

    ```
    clearKeys=node,bizdoc,sender,receiver
    ```

    If the service is receiving a flat file document, to clear the pipeline of *ffdata*, *bizdoc*, *sender*, and *receiver* for this service, specify:

    ```
    clearKeys=ffdata,bizdoc,sender,receiver
    ```

    If the service is receiving an EDI document, to clear the pipeline of *edidata*, *bizdoc*, *sender*, and *receiver* for this service, specify:

    ```
    clearKeys=edidata,bizdoc,sender,receiver
    ```

- If you are invoking this service from a Java program, in addition to returning *bizdoc*, *sender*, and *receiver* as IS documents (IData objects), the service returns *bizdoc* as an instance of com.wm.app.tn.doc.BizDocEnvelope and the returned *sender* and *receiver* as instances of com.wm.app.tn.profile.ProfileSummary.

- The *TN_parms*/*$bypassRouting* variable takes precedent over the **Processing Rule Routing** settings within the TN document type. For example, if the *$bypassRouting* variable is set to `true` to disable processing rule routing, but the TN document type **Processing Rule Routing** settings enable processing rule routing, the *$bypassRouting* variable takes precedent and Trading Networks will bypass processing rule routing.

wm.tn:reroute

Locates a document that you specify in the Trading Networks database and processes it again. To process the document again, Trading Networks looks up the appropriate processing rule and performs the processing actions defined in the processing rule.

Input Parameters

internalId	**String** The internal document ID of the document to reprocess. This is a unique ID that Trading Networks assigns to the document.

Output Parameters

bizdoc	**Document** The document that was processed again. For the structure of *bizdoc*, see wm.tn.rec:BizDocEnvelope.
sender	**Document** The profile summary for the sender of the document. For the structure of *sender*, see wm.tn.rec:ProfileSummary.
receiver	**Document** The profile summary for the receiver of the document. For the structure of *receiver*, see wm.tn.rec:ProfileSummary.
TN_parms	**Document** (optional) An IS document (IData object) that holds *hints* that Trading Networks uses when performing document recognition for a flat file document. For information about document gateway services hints, see *webMethods Trading Networks Administrator's Guide*.

Usage Notes

- If you are invoking this service from a Java program, in addition to returning *bizdoc*, *sender*, and *receiver* as IS documents (IData objects), the service returns *bizdoc* as an instance of com.wm.app.tn.doc.BizDocEnvelope and the returned *sender* and *receiver* as instances of com.wm.app.tn.profile.ProfileSummary.
- Because Trading Networks has already processed the document, it does *not* perform the preprocessing actions again. That is, even if instructed to do so by the TN document type and/or processing rule, Trading Networks does not verify the digital signature of the document, validate the structure of a document, verify if Trading Networks has received the document, or save the document to the database.

wm.tn:submit

Submits a document that has already been recognized to Trading Networks for processing.

This service ensures that the sender of the document matches the current user. If you are sending documents from within processing rules or services and this identity check may fail, see wm.tn.route:routeBizdoc.

Input Parameters

bizdoc	**Object** The recognized document that you want Trading Networks to process. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.

Output Parameters

bizdoc	**Object** The document formatted as an IS document type (IData object). For the structure of *bizdoc*, see wm.tn.rec:BizDocEnvelope.
sender	**Document** The profile summary for the sender of the document. For the structure of *sender*, see wm.tn.rec:ProfileSummary.
receiver	**Document** The profile summary for the receiver of the document. For the structure of *receiver*, see wm.tn.rec:ProfileSummary.
TN_parms	**Document** (optional) An IS document (IData object) that holds "hints" that Trading Networks uses when performing document recognition for a flat file document. See information about document gateway services in *webMethods Trading Networks Administrator's Guide* for details on providing recognition hints.

Usage Notes

- If you are invoking this service from a Java program, in addition to returning *bizdoc*, *sender*, and *receiver* as IS documents (IData objects), the service returns *bizdoc* as an instance of com.wm.app.tn.doc.BizDocEnvelope and the returned *sender* and *receiver* as instances of com.wm.app.tn.profile.ProfileSummary.
- This service returns after Trading Networks completes processing for the document after Trading Networks executes the pre-processing and processing actions for the document. If the processing actions instruct Trading Networks to execute a service asynchronously, the asynchronously invoked service may not be complete.

2 Admin Folder

Overview

Use administrative services (services in the wm.tn.admin folder) to:

- Export information from and import information to the Trading Networks database.
- Retrieve the settings for all defined Trading Networks properties.
- Set any or all of the Trading Networks properties.
- Test the JDBC connection properties.

Summary of Elements in this Folder

The elements that are available in this folder are listed in the following table:

Element	Description
wm.tn.admin:exportData	Exports data from the Trading Networks database.
wm.tn.admin:extendedExportData	Exports data from the Trading Networks database by either saving the data to an export file or by generating sources for solution deployment through webMethods Deployer. Provides extensions for each asset type and an option to filter the asset type based on internal IDs.
wm.tn.admin:extendedImportData	Imports data from the supplied XML or binary file containing Trading Networks data by either saving the data from an export file or by generating sources for solution deployment through webMethods Deployer. Provides extensions for each asset type and an option to filter the asset type based on internal IDs.
wm.tn.admin:getDBLimits	Retrieves the limits set for lengths of all the columns of the Trading Networks database.
wm.tn.admin:getDBPoolInfo	Retrieves the run-time information about the JDBC connection pool associated with the Trading Networks database.
wm.tn.admin:getProperties	Retrieves the Trading Networks properties.
wm.tn.admin:getStartupErrors	Returns the error count and the list of errors that occur during Trading Networks startup.
wm.tn.admin:importData	Imports data into the Trading Networks database.
wm.tn.admin:setProperties	Sets the Trading Networks properties. You can add, update, or delete server properties.

wm.tn.admin:exportData

Exports data from the Trading Networks database.

Input Parameters

attribs **String** (optional) Indicates whether to export document attributes. Valid values are:

- `true` - Export document attributes.
- `false` - Do not export document attributes.

types **String** (optional) Indicates whether to export TN document types. Valid values are:

- `true` - Export TN document types.
- `false` - Do not export TN document types.

rules **String** (optional) Indicates whether to export processing rules. Valid values are:

- `true` - Export processing rules.
- `false` - Do not export processing rules.

flddefs **String** (optional) Indicates whether to export profile field definitions. Valid values are:

- `true` - Export profile field definitions.
- `false` - Do not export profile field definitions.

profile **String** (optional) Indicates whether to export partner profiles. Valid values are:

- `true` - Export partner profiles.
- `false` - Do not export partner profiles.

lookups **String** (optional) Indicates whether to export profile lookup data: ID types, contact types, and binary types. Valid values are:

- `true` - Export lookup data.
- `false` - Do not export lookup data.

tpas **String** (optional) Indicates whether to export Trading Partner Agreements (TPAs). Valid values are:

- `true` - Export TPAs.

- `false` - Do not export TPAs.

extflds **String** (optional) Indicates whether to export the extended fields. Valid values are:

- `true` - Export extended fields.
- `false` - Do not export extended fields.

securityData **String** (optional) Indicates whether to export security data. Valid values are:

- `true` - Export security data.
- `false` - Do not export security data.

queues **String** (optional) Indicates whether to export queues. Valid values are:

- `true` - Export queues.
- `false` - Do not export queues.

dls **String** (optional) Indicates whether to export data permissions. Valid values are:

- `true` - Export data permissions.
- `false` - Do not export data permissions.

fp **String** (optional) Indicates whether to export general functional permissions. Valid values are:

- `true` - Export general functional permissions.
- `false` - Do not export general functional permissions.

archSvcs **String** (optional) Indicates whether to export archived services. Valid values are:

- `true` - Export archived services.
- `false` - Do not export archived services.

all **String** (optional) Indicates whether to export all the Trading Networks objects. Valid values are:

- `true` - Exports attributes, document types, processing rules, profile field definitions, partner profiles, ID types, contact types, binary types, TPAs, extended fields, security data, queues, DLS, functional permissions, and archived services.
- `false` - Do not export all Trading Networks objects. Export only the objects selected.

Output Parameters

data **Document** An IS document (IData object) that contains the exported data. The document contains the following keys:

- *version* **Document** Version information from the Trading Networks database. The document contains the following keys.
  - *major* **String** The major Trading Networks release number.
  - *minor* **String** The minor Trading Networks release number.
- *attribs* **Document** A set of attributes. For each attribute, the key is the *attributeId* in a string and the value is a com.wm.app.tn.doc.BizDocAttribute.
- *types* **Document** A set of TN document types. For each type, the key is the *bizdocTypeId* in a string and the value is a com.wm.app.tn.doc.BizDocType.
- *rules* **Object** A com.wm.app.tn.route.RoutingRuleList. This is the complete set of processing rules for Trading Networks.
- *fldgrps* **Document** A set of field groups. For each field group, the key is the group description in a string and the value is the group code in a short.
- *flddefs* **Object** A java.util.Vector. Each field definition is a com.wm.app.tn.profile.ProfileFieldMetaData.
- *profiles* **Object** A java.util.Vector. Each profile is a com.wm.app.tn.profile.Profile.
- *lookups* **Object** A set of lookup data: profile groups, ID types, contact types, and binary types.
- *profileGroups* **Object** A set of profile groups.
- *extflds* **Object** A set of extended fields.
- *securityData* **Object** A set of security data.
- *queues* **Object** A set of queues.
- *dls* **Object** A set of data level securities.
- *fp* **Object** A set of functional permissions.
- *archSvcs* **Object** A set of archived services.
- *idTypes* **Document** A set of ID types. For each ID type, the key is the type description in a string and the value is the type code in an integer.

- *contactTypes* **Document** A set of contact types. For each contact type, the key is the type description in a string and the value is the type code in an integer.
- *binaryTypes* **Document** A set of binary types. For each binary type, the key is the type description in a string and the value is the type code in an integer.
- *tpas* **Object** A java.util.Vector where each element in the Vector is a Trading Partner Agreement (TPA). Each TPA is a com.wm.app.tn.tpa.TPA.
- *dependency* **Document** A set of dependent assets of exported data.

Usage Notes

- Use the wm.tn.admin:exportData and wm.tn.admin:importData services to transfer data from one Trading Networks database to another. You cannot transfer *all* data in the database using these services. To transfer *all* data, use the appropriate database vendor-supplied utility. The wm.tn.admin:exportData and wm.tn.admin:importData services are useful for copying configuration data, such as, TN document types, attributes, processing rules, and profile field definitions. You cannot use the services to copy operational data, such as, document instances and activity log entries.
- The wm.tn.admin:exportData and wm.tn.admin:importData services are intended to be used together. The structure of the *data* output from wm.tn.admin:exportData service matches the structure of the *data* input variable for wm.tn.admin:importData service.

wm.tn.admin:extendedExportData

Exports data from the Trading Networks database by either saving the data to an export file or by generating sources for solution deployment through webMethods Deployer. Provides extensions for each asset type and an option to filter the asset type based on internal IDs.

Input Parameters

type **String** Indicates whether to save the export data in binary format or XML format. Valid values are:

- `bin` - Save the export data in binary format.
- `xml` - Save the export data in XML format.

all **String** Specifies which Trading Networks assets to export. Valid values are:

- `true` - Export all Trading Networks assets.
- `false` - Export only those Trading Networks assets that are configured in the *exportData* document, described below.

acdl **String** Indicates whether to export the Trading Networks assets as sources for solution deployment using webMethods Deployer. Valid values are:

- `true` - Export the assets as sources for solution deployment using webMethods Deployer.
- `false` - Export the assets as a Trading Networks XML or binary export file.

exportData **Document List** An IS document (IData object) containing the TPA data being passed to a validation service for validation. The document contains the following keys:

- *assetType* **String** Indicates which asset type to include in the export. Valid values are `documenttype`, `documentattribute`, `processingrule`, `partner`, `fieldgroup`, `fielddefinition`, `externalidtype`, `contacttype`, `binarytype`, `profilegroup`, `queue`, `tpa`, `dls`, `functionalpermission`, `extendedfield`, and `archiveschedule`.
- *all* **String** Indicates whether to export all Trading Networks assets to the export file. Valid values are:
  - `true` - Export all assets of the specified asset type.
  - `false` - Export only assets of a specified asset type that match the internal IDs specified in the *ids* variable.
- *ids* **String List** Specifies the internal IDs associated with the assets to export. For the queue and archive schedule asset types, the internal ID is considered the name of the asset; for all other asset types, the internal ID is the individual asset ID.

exportFileDirectory **String** (optional) Indicates the output directory for the export file or solution package. The directory that you specify must already exist. If you do not specify a export directory path, Trading Networks creates the export file or solution package in the directory specified by the `tn.tmpdir` property in the TN properties file.

exportFileName **String** Indicates the name of the export file or solution package. If you do not specify a name, Trading Networks names the file "export."

Output Parameters

output **Document** Contains the results of the export. The document contains the following keys:

- *exportFilePath* **String** Path of the zip file containing the exported assets. The location of this file is determined by the input parameter *exportFileDirectory* or `tn.tmpdir` property in the TN properties file.

- *errorMessages* **String List** Any errors encountered during the export process.

Usage Notes

- If the assets are exported as a file rather than as a solution package, Trading Networks creates a zip file containing the XML or binary export file. When extracted, the contents of this zip file can be used for import using My webMethods or a custom service.
- If you do not specify *exportFileDirectory*, Trading Networks creates the export file or solution package in the directory path specified by the `tn.tmpdir` property in the TN properties file.

wm.tn.admin:extendedImportData

Imports data from the supplied XML, binary, or zip file containing Trading Networks data by either saving the data from an export file or by generating sources for solution deployment through webMethods Deployer. Provides extensions for each asset type and an option to filter the asset type based on internal IDs

Input Parameters

force **String** If the keys in the *data* variable match the keys in the database, this variable indicates whether the rows in the database should be overwritten. Valid values are:

- `true` - Overwrite rows in the database if the keys from the *data* variable match.
- `false` - Does not overwrite rows in the database if the keys from the *data* variable match.

type Indicates whether the file that contains data is a binary, XML, or a zip file.

replaceRuleList **String** (Optional) Indicates whether the processing rules from the supplied data should replace the list of processing rules in the database. Valid values are:

- `true` - Replace all of the processing rules in the database with those in *data*.
- `false` - Append the processing rules in *data* to the rules in the database. Default.

all **String** Specifies which Trading Networks assets to import. Valid values are:

- `true` - Import all Trading Networks assets.

- `false` - Import only those Trading Networks assets that are configured in the *importData* document, described below.

importDataFileName **String** Indicates the name of the file from which data is to be imported.

importData **Document** The data to import into the Trading Networks database. The document contains the following keys:

- *assetType* **String** Indicates which asset type to import. Valid values are `documenttype`, `documentattribute`, `processingrule`, `partner`, `fieldgroup`, `fielddefinition`, `externalidtype`, `contacttype`, `binarytype`, `profilegroup`, `queue`, `tpa`, `dls`, `functionalpermission`, `extendedfield`, and `archiveschedule`.
- *all* **String** Indicates whether to import all Trading Networks assets of a specified asset type. Valid values are:
  - `true` - Import all assets of the specified asset type.
  - `false` - Import only assets of a specified asset type that match the internal IDs specified in the *ids* variable.
- *ids* **String List** Specifies the internal IDs associated with the assets to import. For the queue and archive schedule asset types, the internal ID is considered the name of the asset; for all other asset types, the internal ID is the individual asset ID.

version **Document** Version information from the Trading Networks database. The document contains the following keys:

- *major* **String** The major Trading Networks release number.
- *minor* **String** The minor Trading Networks release number.

Output Parameters

output **Document** Contains the results of the import. The document contains the following keys:

- *assets* **Document List** The data that was imported into the Trading Networks database. The document contains the following keys:
  - *assetType* **String** The asset type that was imported.
  - *id* **String** The internal IDs associated with the asset.
  - *name* **String** The name of the asset.
  - *status* **String** The outcome of the import of the asset (`success` or `fail`).

- *message* **String** The status message from the last attempt to import the asset.
- *stacktrace* **String** The stack trace of the exception if the import failed.

- *errorMessages* **String List** Any errors encountered during the import process.

wm.tn.admin:getDBLimits

Retrieves the limits set for lengths of all the columns of the Trading Networks database.

Input Parameters

None.

Output Parameters

limits	**Object** A java.lang.Hashtable that contains the lengths of all the columns of the Trading Networks database. The Hashtable is keyed by the *String* table name and column name combination (TableName.ColumnName), and the values are the *Integer* lengths of the columns.

Usage Notes

- The Trading Networks database column length limits are stored in the Trading Networks dblimits file (dblimits.cnf). The Trading Networks dblimits file is located in the WmTN package in the directory, *Integration Server_directory* \instances*instance_name*\packages\WmTN\config\dblimits.cnf.

wm.tn.admin:getDBPoolInfo

Retrieves the run-time information about the JDBC connection pool associated with the Trading Networks database.

Input Parameters

None.

Output Parameters

poolName	**String** (optional) Indicates the name of the JDBC connection pool.
status	**String** (optional) Indicates whether the service was able to connect the JDBC connection pool. Valid values are:

- `Active` Indicates that the service was able to connect to the JDBC pool.
- `Inactive` Indicates that the service was able to connect to the JDBC pool.

maxConns	**String** (optional) Indicates the maximum number of connections the pool can have.
minConns	**String** (optional) Indicates the minimum number of connections the pool can have.
poolSize	**String** (optional) Indicates the number of connections that exist in the pool.
availableConns	**String** (optional) Indicates the number of connections that are available for use.
idleTimeout	**String** (optional) Number of milliseconds the service will wait to obtain a connection to a JDBC connection pool.

wm.tn.admin:getProperties

Retrieves the Trading Networks properties.

Input Parameters

None.

Output Parameters

props	**Document** The settings in the Trading Networks properties. The variable names in *props* are the names of the Trading Networks properties from the Trading Networks properties file. All values in *props* have the data type String.

Usage Notes

- The Trading Networks properties (which all start with "tn") are stored in the Trading Networks properties file (properties.cnf). The Trading Networks properties file is located in the WmTN package in the directory, *Integration Server_directory* \instances*instance_name*\packages\WmTN\config\properties.cnf.
- For a complete list of the Trading Networks properties, view the online help files that you access from the TN Properties page. To access this help from the Integration Server Administrator, select **Trading Networks** from the **Solutions** menu of the navigation panel. Trading Networks displays the TN Properties page. In the upper right corner of the TN Properties page, click **Help**.

wm.tn.admin:getStartupErrors

Returns the error count and the list of errors that occur during Trading Networks startup.

Input Parameters

None.

Output Parameters

DBErrorCount	**String** Indicates the number of database errors that occurred during Trading Networks startup.
DBErrors	**String** (optional) List of database errors that occurred during Trading Networks startup.
ErrorCount	**String** Indicates the number of general errors (other than database errors) that occurred during Trading Networks startup.
Errors	**String** (optional) List of general errors that occurred during Trading Networks startup.

wm.tn.admin:importData

Imports data into the Trading Networks database.

Input Parameters

force

String If the keys in the *data* variable match the keys in the database, this variable indicates whether the rows in the database should be overwritten. Valid values are:

- `true` - Overwrite rows in the database if the keys from the *data* variable match.
- `false` - Do not overwrite rows in the database if the keys from the *data* variable match.

overwriteRules

String (optional) Indicates whether the ProcessingRules from the supplied data should replace the list of ProcessingRules in the database. Valid values are:

- `true` - Replace all the ProcessingRules in the database with those in *data*.
- `false` - Default. Append the ProcessingRules in *data* to the rules in the database.

data **Document** The data to import into the Trading Networks database. The document contains the following keys:

- *version* **Document** Version information from the Trading Networks database. The document has the following keys.
  - *major* **String** The major Trading Networks release number.
  - *minor* **String** The minor Trading Networks release number.
  - *force* **String** If the values for *major* or *minor* differ from the major and minor version in the database. *Force* indicates whether to overwrite the version information in the database. Valid values are:
    - `true` - Overwrite version
    - `false` - Do not overwrite version
- *attribs* **Document** (optional) A set of attributes to import. For each attribute, the key is the *attributeId* in a string and the value is a com.wm.app.tn.doc.BizDocAttribute.
- *types* **Document** (optional) A set of TN document types to import. For each type, the key is the *bizdocTypeId* in a string and the value is a com.wm.app.tn.doc.BizDocType.
- *rules* **Object** (optional) A set of processing rules to import. Specify a com.wm.app.tn.route.RoutingRuleList for *rules*. This is the complete set of processing rules for Trading Networks.

  If you set *overwriteRules* to `false` and the database already has processing rules, this service does not import the data specified in *rules*.
- *fldgrps* **Document** (optional) A set of field groups. For each field group, the key is the group description in a string and the value is the group code in a short.
- *flddefs* **Object** (optional) A java.util.Vector. This is a set of profile field definitions to import. Each field definition is a com.wm.app.tn.profile.ProfileFieldMetaData.
- *profiles* **Object** (optional) A java.util.Vector. This is a set of partner profiles to import. Each profile is a com.wm.app.tn.profile.Profile.
- *idTypes* **Document** (optional) A set of ID types. For each ID type, the key is the type description in a string and the value is the type code in an integer.
- *contactTypes* **Document** (optional) A set of contact types. For each contact type, the key is the type description in a string and the value is the type code in an integer.

- *binaryTypes* **Document** (optional) A set of binary types to import. For each binary type, the key is the type description in a string and the value is the type code in an integer.
- *tpas* **Object** (optional) A java.util.Vector. This is a set of trading partner agreements to import. Each trading partner agreement is a com.wm.app.tn.tpa.TPA.
- *extflds* **Object** (optional) A java.util.Vector. This is a set of extended fields to import.
- *profileGroups* **Object** (optional) A java.util.Vector. This is a set of partner profile groups.
- *securityData* **Object** (optional) A java.util.Vector. This is security data set.

Output Parameters

errors **Document** Exceptions that occur while importing the contents of data are returned in *errors*. The document contains the keys:

- *attribs* **Document** (optional) Exceptions with importing document attributes. For each exception, the key is the *attributeId* and the value is an IS document (IData object).
- *types* **Document** (optional) Exceptions with importing TN document attributes. For each exception, the key is the *bizdocTypeId* and the value is an IS document (IData object).

  Note:
  If using an OEM version of Trading Networks, you cannot add or import new TN document types.
- *rules* **Document** (optional) Exceptions with importing processing rules. The key is *rules* and the value is an IS document (IData object).
- *fldgrps* **Document** (optional) Exceptions with importing field groups. For each exception, the key is the field group description and the value is an IS document (IData object).
- *flddefs* **Document** (optional) Exceptions with importing profile field definitions. For each exception, the key is the *profileFieldID* and the value is an IS document (IData object).
- *profiles* **Document** (optional) Exceptions with importing partner profiles. For each exception, the key is the *partnerId* and the value is an IS document (IData object).

- *idTypes* **Document** (optional) Exceptions with importing ID types. For each exception, the key is the ID type description and the value is an IS document (IData object).
- *contactTypes* **Document** (optional) Exceptions encountered importing contact types. For each exception, the key is the contact type description and the value is an IS document (IData object).
- *binaryTypes* **Document** (optional) Exceptions with importing binary types. For each exception, the key is the binary type description and the value is an IS document (IData object).
- *queues* **Document** (optional) Exceptions with importing queues. For each exception, the key is the ID of the queue and the value is an IS document (IData object).
- *tpas* **Document** (optional) Exceptions with importing TPAs. For each exception, the key is the TPA ID and the value is an IS document (IData object).
- *extflds* **Document** (optional) Exceptions with importing extended fields. For each exception, the key and the value is an IS document (IData object).
- *profileGroups* **Document** (optional) Exceptions with importing the partner profile groups. For each exception, the key is the ID of the profile group and the value is an IS document (IData object).
- *securityData* **Document** (optional) Exceptions with importing the security data. For each exception, the key is the ID of the security data and the value is an IS document (IData object).

Usage Notes

- Use the wm.tn.admin:exportData and wm.tn.admin:exportData services to transfer data from one Trading Networks database to another. You cannot transfer *all* data in the database using these services. To transfer *all* data, use the appropriate database vendor-supplied utility. The wm.tn.admin:exportData and wm.tn.admin:exportData services are useful for copying configuration data, such as, TN document types, attributes, processing rules, and profile field definitions. You cannot use the services to copy operational data, such as, document instances and activity log entries.
- The wm.tn.admin:exportData and wm.tn.admin:exportData services are intended to be used together. The structure of the output from wm.tn.admin:exportData service matches the structure of the *data* input variable for wm.tn.admin:exportData service.
- If you are using an OEM version of the Trading Networks, you cannot import new TN document types. This service will fail in an OEM environment.

wm.tn.admin:setProperties

Sets the Trading Networks properties. You can add, update, or delete server properties.

Input Parameters

props	**Document** The Trading Networks properties to add or update. For the variable names in *prop*, specify the names of the Trading Networks properties to set. For the values, specify the values to assign each property. All variables in *prop* should have the data type string. You can specify any of the Trading Networks properties.
deletedProps	**String List** The properties to delete from the Trading Networks properties file.

Output Parameters

updateCount	**String** The number of properties that Trading Networks added, updated, and deleted.

Usage Notes

- The Trading Networks properties (which start with "tn") are stored in the Trading Networks properties file (properties.cnf). The Trading Networks properties file is located in the WmTN package in the directory, *Integration Server_directory* \instances*instance_name*\packages\WmTN\config\properties.cnf.
- For a complete list of the Trading Networks properties, view the online help files that you access from the TN Properties page. To access this help, from the Integration Server Administrator, click **Trading Networks** from the **Solutions** menu of the navigation panel. Trading Networks displays the TN Properties page. In the upper right corner of the TN Properties page, click **Help**.

3 Archive Folder

Overview

Use archiving services (services in the wm.tn.archive folder) to manage your database; that is, to conserve disk space and consolidate bookkeeping of old documents.

The wm.tn.archive:archive and the wm.tn.archive:extendedArchive services archive documents or document information by removing the information and the references to the document from all production tables, and placing it in the corresponding archival tables. When a document is archived, using My webMethods you can view or perform tasks on it just like you do on production data.

The wm.tn.archive:archive service deletes documents from both, archival and production tables. However, the wm.tn.archive:extendedArchive service can delete documents or document information from either the archival tables, production tables, or both.

For more information about archiving and deleting documents, see *webMethods Trading Networks Administrator's Guide*.

Summary of Elements in this Folder

The elements that are available in this folder are listed in the following table:

Element	Description
wm.tn.archive:archive	Archives documents and deletes documents from the Trading Networks database.
wm.tn.archive:extendedArchive	Archives documents and deletes documents from the Trading Networks database, based on the criteria that you specify.
wm.tn.archive:archiveByStoredProc	Archives documents and deletes documents from the Trading Networks database using the stored procedure based archive script.
wm.tn.archive:purgeBizdocUniqueKeysData	Purges data from BizDocUniqueKeys and ARCHIVE_BizDocUniqueKeys tables.
wm.tn.archive:purgeEDITrackingData	Purges data from EDITracking and ARCHIVE_EDITracking tables.

wm.tn.archive:archive

Archives documents and deletes documents from the Trading Networks database.

Input Parameters

archiveAfterDays **String** (optional) The maximum number of days to store a document in the production tables (since being received) before being archived. This

service archives documents that have been in the database longer than the number of days you specify.

Specify a value from 0 through 730365. 0 indicates to not archive documents.

If you do not specify a value for *archiveAfterDays*, this service uses the setting of the `tn.archive.archiveAfterDays` property in the Trading Networks properties file located in the directory, *Integration Server_directory* \instances*instance_name*\packages\WmTN\config\properties.cnf.

deleteAfterDays **String** (optional) The maximum number of days that a document is to remain in the database (since it was received) before being deleted. This service deletes documents that have been in the database longer than the number of days you specify.

Specify a value from 0 through 730365. If you specify 0, Trading Networks does not delete documents.

If you do not specify a value for *deleteAfterDays*, this service uses the setting of the `tn.archive.deleteAfterDays` property in the Trading Networks properties file. The Trading Networks properties file is in the directory, *Integration Server_directory* \instances*instance_name*\packages\WmTN\config\properties.cnf.

Output Parameters

archiveCount **String** The number of documents that the service archived.

deleteCount **String** The number of documents that the service deleted from the database.

When you run services such as wm.tn.archive:archive or wm.tn.archive:extendedArchive:

1. For each batch, based on the Trading Networks property tn.archive.batchSize, the document IDs to be archived/deleted are copied to the ARCHIVE_WorkTable.
2. The document IDs are looped over and deleted.
3. Once the actual records are deleted, the records are deleted from the ARCHIVE_WorkTable.
4. If, for some reason, the job does not terminate normally, for example, the Oracle rollback segment overflows due to a large batch size, there might be leftover records in ARCHIVE_WorkTable. If this happens, these records have to be deleted before attempting another archive operation.

Starting Trading Networks versions 10.1 or higher, archive stored procedures were introduced to streamline the process. You can manually delete the records by date, but this can be problematic with a large number of records due to the dependency on rollback segment sizing.

The tn.archive.batchSize determines the number of records to commit per transaction, making the behavior easier to control.

Usage Notes

- You can execute this service from time to time to conserve space in the system database. You can use Integration Server Administrator to schedule a user task to automatically execute this service periodically. You may not need this service for sites with large databases and that have one or more database administrators because such sites usually have their own archiving constraints.
- If you do not specify *archiveAfterDays* and the `tn.archive.archiveAfterDays` is not set in the Trading Networks properties file, the service deletes documents, but does not archive any documents.
- If the setting that controls deleting documents (either *deleteAfterDays* specified with the service or the `tn.archive.deleteAfterDays`) is less than the setting that controls archiving documents (either *archiveAfterDays* specified with the service or the `tn.archive.archiveAfterDays` property), this service does not archive files, only deletes them.

wm.tn.archive:extendedArchive

Archives documents and deletes documents from the Trading Networks database, based on the criteria that you specify.

Input Parameters

operation	**String** The operation to perform. Specify `archive` or `delete`.
deletionType	**String** Whether to delete data from archival or production tables, or both. Valid values are: - `Archival` Default. Delete documents from archive tables. - `Production` Delete documents from production tables. - `Both` Delete documents from both archive and production tables.
afterDays	**String** (optional) The maximum number of days after which to archive or delete stored documents. Specify a value from 0 through 730365. 0 indicates to not archive or delete documents.
backOffTime	**String** (optional) The number of seconds that Trading Networks must wait between two batches of archive or delete during one schedule. The default time is 15 seconds. You can also set the default value using the `tn.archive.batchBackoffTime` property in the properties.cnf file. For information about the property, see *webMethods Trading Networks Administrator's Guide*.

maxRows	**String** (optional) The maximum number of documents that Trading Networks can archive or delete during one schedule. Trading Networks archives or deletes documents in batches. For example, if *batchSize* is set to 50 and the *maxRows* is set to 1000, then Trading Networks performs the archive or delete 20 times during that schedule. There is no limit on the maximum number of documents per schedule. However, the value specified for *maxRows* must be greater than or equal to the *batchSize* value. When the total number of documents for archive or deletion is more than the maximum number of documents per schedule, Trading Networks attempts to archive or delete the additional documents during the next schedule. You can set the default value using the `tn.archive.maxRows` property in the properties.cnf file. For information about the property, see *webMethods Trading Networks Administrator's Guide*.
batchSize	**String** (optional) The maximum number of documents that can be archived or deleted in a batch. The default is 100 documents. Set the default value in the `tn.archive.batchSize` property defined in the properties.cnf file. For information about this property, see *webMethods Trading Networks Administrator's Guide*.
docTypeId	**String** (optional) For XML documents, specify the DOCTYPE identifier, which can be either the system identifier or public identifier within the XML document. These identifiers are located in the document type declaration (DOCTYPE) after either the "SYSTEM" or "PUBLIC" literal string. For flat file documents, specify the Trading Networks-generated internal identifier of the TN flat file document type. To determine the document type identifier invoke the wm.tn.doctype:list service from Software AG Designer to return the name and ID of all your TN document types.
senderId	The internal ID specified in the sender's profile. If you do not specify any value, Trading Networks considers the documents of all senders for the archival or deletion.
receiverId	The internal ID specified in the receiver's profile. If you do not specify any value, Trading Networks considers the documents of all receivers for the archival or deletion.
systemStatus	The status of a document after Trading Networks has processed it, for example `DONE` or `DONE W/ERRORS`.
userStatus	A value that a processing rule assigned to the document's **User Status** system attribute, for example, `accepted`, `rejected`, or `pending approval`.

options **String** Whether to archive or delete any of the following:

- Bizdoc
- Document content
- Custom attributes
- Custom array attributes
- Related documents
- Delivery tasks
- Activity log entries
- Unique keys
- EDI tracking details

The following values indicate:

- `true` - Archive or delete the document information.
- `false` - Do not archive or delete the document information.

Output Parameters

count **String** Optional. The number of documents archived or deleted from the production database.

This parameter does not provide the count for documents deleted from the archive database. For example, if you execute the service to delete documents from the archive database, the count is `0` (zero) because nothing is deleted from the production database.

Usage Notes

- When *operation* is:
  - `Delete`, BizDoc (under options) is `true`, and regardless of whether the other child tables of BizDoc are `true` or `false`, all the child records are deleted.
  - `Delete`, BizDoc (under options) is `false`, the corresponding child records are deleted for all the child tables that are `true`.
  - `Archive`, BizDoc (under options) is `true`, and all the other child tables that are true are archived and those that are `false` are deleted.
- You can troubleshoot the status of the delete or archive task by checking the corresponding entries in Integration Server logs. Information such as the number of documents that will be

archived or deleted, batch size, and number of batches completed are captured in Integration Server logs.

An example of Integration server log entries for an archive or delete operation with 9600 transaction, in two batches, with batch size 5000 is as follows:

```
YYYY-MM-DD 12:34:19 EDT [TNS.0000.1011W] Archive: Starting BizDoc
Delete operation.
YYYY-MM-DD 12:34:19 EDT [TNS.0000.1011W] Archive: Expecting to
delete 9600 transactions (with batchSize=5000), will require
2 batches.
YYYY-MM-DD 12:34:58 EDT [TNS.0000.1011W] Archive: Completed Batch
1 of 2.
YYYY-MM-DD 12:35:47 EDT [TNS.0000.1012W] Archive: Completed Batch
2 of 2.
YYYY-MM-DD 12:35:47 EDT [TNS.0000.1012W] Archive: Finished BizDoc
Delete operation. Returned count=9600
```

wm.tn.archive:archiveByStoredProc

Archives documents and deletes documents from the Trading Networks database using the stored procedure based archive script.

Input Parameters

operation

String The operation to perform. The available options are `archive` or `delete`.

Value	Description
Archive	Moves the data from the production table to the archive table.
DeleteArchived	Deletes the data from the archive table.
DeleteProduction	Deletes the data from the production table.

afterDays

String The maximum number of days after which the stored documents are archived or deleted.

Specify any value for *afterDays* from `1` to `96665`.

Note:
afterDays parameter supports only integers.

batchCount

String (optional) The maximum number of documents that can be archived or deleted in a batch.

The default is `1000` documents. Set the default value in the `tn.archive.batchSize` property defined in the `properties.cnf` file. For information about this property, see *webMethods Trading Networks Administrator's Guide*.

Note:
batchCount parameter supports only integers.

Output Parameters

numberOfRowsImpacted	**String** The number of rows of data that the service successfully archives or deletes from the production table or deletes from the archive table.
errors	**String List** (optional) Retrieves all the errors resulted from the archive or delete operation.

wm.tn.archive:purgeBizdocUniqueKeysData

Purges data from BizDocUniqueKeys and ARCHIVE_BizDocUniqueKeys tables.

Input Parameters

maxRows	**String** (optional) The maximum number of rows that Trading Networks should purge from the tables. Trading Networks purges data in batches. For example, if *batchSize* is set to 50 and the *maxRows* is set to 1000, then Trading Networks purges 20 times during that schedule.
batchSize	**String** (optional) The maximum number of rows of data that can be purged in a batch.

Output Parameters

count	**String** The number of rows of data that the service purged from BizDocUniqueKeys and ARCHIVE_BizDocUniqueKeys tables.

wm.tn.archive:purgeEDITrackingData

Purges data from EDITracking and ARCHIVE_EDITracking tables.

Input Parameters

maxRows	**String** (optional) The maximum number of rows that Trading Networks should purge from the tables.

	Trading Networks purges data in batches. For example, if *batchSize* is set to 50 and the *maxRows* is set to 1000, then Trading Networks purges 20 times during that schedule.
batchSize	**String** (optional) The maximum number of rows of data that can be purged in a batch.

Output Parameters

count	**String** The number of rows of data that the service purged from EDITracking and ARCHIVE_EDITracking tables.

4 Charting Folder

Overview

Use the charting service (service in the wm.tn.charting folder) to manage your dashboard tables, and to conserve disk space and purge old data. The dashboard tables include:

- TransactionSummaryData
- CustomAttributeVolumeValue
- TransactionSuccessFailedData
- SuccessFailedChartDocIdMap
- TransactionLateFAData

Summary of Elements in this Folder

The elements that are available in this folder are listed in the following table:

Element	Description
wm.tn.charting:purgeTransactionSummaryData	Purges records from the dashboard tables.
wm.tn.charting:populateSummaryData	During migration, purges existing records from the dashboard tables, and populates records in the dashboard tables from runtime tables.

wm.tn.charting:purgeTransactionSummaryData

Purges records from the dashboard tables.

Input Parameters

afterDays

String The time frame to consider before purging records from the dashboard tables. This service purges records starting from the first record up to all records present in the table *n* days prior to the current day, where *n* is the number of days you specify for the parameter.

For example, if `afterDays = 5`, this service purges all records starting from the first record up to all records present in the table `5` days prior to the current day. For more details about purging records, see "Usage Notes" on page 47.

Output Parameters

success **String** Whether the records are purged. Valid values are:

- `true` - Records are successfully purged.
- `false` - Error occurred while purging records from the table.

Usage Notes

If `afterDays = 5`, current date is September 05, 2014, and if the service is run at 10:40 GMT, the service purges all records starting from the first record up to the records present in the table until 10.29 GMT of August 31st, 2014. The records whose timestamp is between 10.30 GMT and 10:40 GMT of August 31st are retained in the table because the purge service purges records on a half hour basis.

wm.tn.charting:populateSummaryData

During migration, purges existing records from the dashboard tables, and populates records in the dashboard tables from runtime tables.

Input Parameters

batchSize **String** The maximum number of records that can be committed in a batch.

For example, if `batchSize = 1000`, this service purges all existing records from the dashboard tables, and populates 1000 records at a time from runtime tables and commits them in the dashboard tables. Default *batchSize* is 100000.

Output Parameters

success **String** Whether records are populated. Valid values are:

- `true` - Records are successfully populated.
- `false` - Error occurred while populating records in the dashboard table.

5 Delivery Folder

Overview

Use the reliable delivery services (services in the wm.tn.delivery folder) for the delivery and tracking of outbound documents between partners.

Before you can use reliable delivery (wm.tn.delivery:deliver) to send an outbound document to a partner, the delivery service for the transfer protocol you want to use must be registered. To deliver the document, invoke the wm.tn.delivery:deliver service with the document and the delivery service as inputs. You can then check the status of the delivery using the wm.tn.task:getTaskStatus. You can check the results of sending the document with the wm.tn.task:getTaskOutput service.

Summary of Elements in this Folder

The elements that are available in this folder are listed in the following table:

Element	Description
wm.tn.delivery:deliver	Delivers a document to a partner in the trading network.
wm.tn.delivery:getRegisteredService	Retrieves a delivery service.
wm.tn.delivery:getRegisteredServices	Retrieves the names of the registered delivery services.
wm.tn.delivery:isServiceRegistered	Determines whether a delivery service is registered.
wm.tn.delivery:refreshServiceCache	Refreshes the delivery service cache within the reliable delivery engine.
wm.tn.delivery:registerDefaults	Registers default delivery services for transport protocols.
wm.tn.delivery:registerService	Registers a delivery service.
wm.tn.delivery:removeService	Unregisters a delivery service.

wm.tn.delivery:deliver

Delivers a document to a partner in the trading network.

Input Parameters

serviceName **String** The name associated with the delivery service to use to deliver the document. (This is not the fully-qualified name of the service. It is the name that was associated with the delivery service when it was registered.)

bizdoc **Document** The document you want to deliver.

	If invoking from a Java program, the document must be an instance of com.wm.app.tn.doc.BizDocEnvelope. Otherwise, *bizdoc* must be in the structure of wm.tn.rec:BizDocEnvelope.
ttw	**String** (optional) If the delivery fails, the number of milliseconds you want the task engine to wait before making its first attempt to redeliver the document. (The task engine uses *ttw* along with *retryFactor* to calculate how long to wait for subsequent retry attempts.)
retryLimit	**String** (optional) If the first attempt to deliver the document fails, the number of additional attempts to retry delivering the document.
retryFactor	**String** (optional) The factor you want task engine to use when determining how long to wait before making the second and subsequent attempts to redeliver the document. The task engine calculates the time to wait by multiplying the last wait time by *retryFactor*. Specify a whole number greater than zero for *retryFactor.*
username	**String** (optional) The user name to use when connecting to a partner's server to delivery the *bizdoc*. If you do not specify *username*, this service uses the user name specified in the partner's profile.
password	**String** (optional) The password (which is associated with *username*) to use when connecting to a partner's server to delivery the *bizdoc*. If you do not specify *username*, this service uses the user name specified in the partner's profile.

Output Parameters

deliveryID	**String** Deprecated. A unique ID that the task engine generates for the delivery task. This output parameter has been deprecated. Use *taskId* instead.
taskId	**String** A unique ID that the task engine generates for the delivery task.
serviceOutput	**Document** (optional) The output that the delivery service returned. The document contains the following keys: ■ *status* - **String** The outcome of the delivery, either `success` or `fail`. ■ *statusMessage* - **String** The status message from the last attempt to deliver the document. For example, if the document is being delivered using HTTP, the status message may be `200 OK`. ■ *output* - **Document** The output that the delivery service returned. ■ *transportTime* - **String** Total time for transporting the document by the delivery service. The *transportTime* is specified in milliseconds.

Usage Notes

- This service uses the profile of the receiving partner identified in the bizdoc to determine *ttw, retryLimit,* and *retryFactor*. If you supply these values in the service input, the values you provide override settings specified in the receiving partner's profile.
- If the document is *not* saved to the Trading Networks database, the task engine is bypassed and Delivery Folder attempts to deliver the document only a single time. In this situation, the *ttw* and *retryLimit* values are not used. The output value *serviceOutput* contains the output of the single attempt to deliver the document. Otherwise, the service returns no output; instead use wm.tn.task:getTaskOutput.
- For backwards compatibility, the output for this service variable *deliveryId* contains the unique ID for the delivery task; that is, the same value that is returned in the *taskId* variable.

wm.tn.delivery:getRegisteredService

Retrieves a delivery service.

Input Parameters

serviceName	**String** The name associated with the delivery service that you want to retrieve. (This is not the fully-qualified name of the service. It is the name that was associated with the delivery service when it was registered.)

Output Parameters

deliveryService	**Document** The delivery service identified by *serviceName*. For the structure of *deliveryService*, see wm.tn.rec:DeliveryService.

Usage Notes

- If *serviceName* is not valid or the delivery service does not exist, the service throws an exception.
- If you are invoking this service from a Java program, in addition to returning *deliveryService* as an IS document (IData object), the service returns *deliveryService* as an instance of com.wm.app.tn.delivery.DeliveryService.

wm.tn.delivery:getRegisteredServices

Retrieves the names of the registered delivery services.

Input Parameters

type **String** Optional - The type of delivery services that you want to retrieve. Valid values are:

- `immediate` - Default. Retrieve immediate delivery services.
- `scheduled` - Retrieve scheduled delivery services.
- `all` - Retrieve all registered delivery services-both immediate and scheduled delivery services.

Output Parameters

services **String List** The names of the registered delivery services.

wm.tn.delivery:isServiceRegistered

Determines whether a delivery service is registered.

Input Parameters

serviceName **String** The name associated with the delivery service that you want to determine is registered or not. (This is not the fully-qualified name of the service. It is the name that was associated with the delivery service when it was registered.)

Output Parameters

registered **String** Whether the delivery service is registered. Valid values are:

- `true` - The specified delivery service is registered.
- `false` - The specified delivery service is *not* registered.

wm.tn.delivery:refreshServiceCache

Refreshes the delivery service cache within the reliable delivery engine.

Input Parameters

None.

Output Parameters

None.

Usage Notes

- Internally, this service is used in a clustered environment to inform all Integration Servers in the cluster to update their delivery service cache after a new delivery service has been added.
- This service is set to disable service redirection. For more information, see pub.cluster:disableServiceRedir, described in *webMethods Integration Server Built-In Services Reference*.

wm.tn.delivery:registerDefaults

Registers default delivery services for transport protocols. For more information about these delivery services, see "Transport Folder" on page 287.

Input Parameters

None.

Output Parameters

None.

Usage Notes

The service is invoked at server start up to check for pre-registered delivery services of the same name as a delivery service (for example, primary HTTP or secondary HTTP) provided with Trading Networks and does not register the default provided delivery services if services of the same name exist.

wm.tn.delivery:registerService

Registers a delivery service.

Input Parameters

serviceName	**String** The name you want to associate with the delivery service.
host	**String** (optional) The host name or IP address of the Integration Server on which to invoke this delivery service. If you do not specify *host*, this services uses "localhost."

port	**String** (optional) The port number that the Integration Server on which to invoke this delivery service listens for incoming requests. If you do not specify *port*, this service uses port number "5555."
user	**String** (optional) The user name to supply when invoking the delivery service.
password	**String** (optional) The password (for the user name specified in *user*) to supply when invoking the delivery service.
ifc	**String** The fully-qualified name of the folder for the delivery service.
svc	**String** The service name for the delivery service.
scheduled	**String** Whether the delivery service is a scheduled delivery service or an immediate delivery service. Valid values are: ■ `true` - For a scheduled delivery service. ■ `false` - For an immediate delivery service.

Output Parameters

None.

Usage Notes

- If you do not supply *host*, the delivery service is invoked directly.
- If you supply a value for *host* other than null or localhost, Trading Networks opens an HTTP connection to that host and invokes this service to deliver a document. If the service actually resides on the localhost, do not supply a host name or IP address. If you do, unnecessary HTTP connections are opened on your Integration Server.

wm.tn.delivery:removeService

Unregisters a delivery service.

Input Parameters

serviceName	**String** The name associated with the delivery service that you want to unregister. (This is not the fully-qualified name of the service. It is the name that was associated with the delivery service when it was registered.)

Output Parameters

None.

Usage Notes

- If the specified delivery service is not registered, this service throws an exception.
- Before using wm.tn.delivery:removeService to remove the delivery service, you *must* delete all delivery tasks that use the delivery service. You can delete a delivery task using the wm.tn.task:removeTask service or from My webMethods. For more information about deleting a delivery task, see *webMethods Trading Networks User's Guide*.

6 Dictionary Folder

Overview

Use the dictionary services (services in the wm.tn.dictionary folder) to create, retrieve, update, and delete profile fields and field groups. In addition, use these services to look up data, for example, contact types, ID types and binary types.

You can use the dictionary services to extend the standard profiles that are provided. The following table lists items that you can extend:

Item	Description
Profile Fields	You can define extended profile fields to extend the information that Trading Networks maintains in profiles beyond the standard fields. A profile includes the standard fields and the extended fields that you define. To define custom fields, use the wm.tn.dictionary:addFieldDefinition service. Trading Networks displays the extended field in profiles in My webMethods for all profiles. For more information about working with field definitions, see *webMethods Trading Networks Administrator's Guide*. For flow programmers, the structure of an extended profile field is defined by the wm.tn.rec:FieldMetaData IS document type. For Java programmers, an extended field definition is a ProfileFieldMetaData object. See the Java API documentation for details.
Field Groups	Each profile field (standard and extended) belongs to a field group. When you define a new extended field, you must specify the group to which the field belongs. Some field groups are provided but the list of field groups is extensible. Use the wm.tn.dictionary:addFieldGroup service to define additional field groups. You can associate extended fields with the field group. You can select the group programmatically using a built-in service, or by using My webMethods.
Contact Types	Two types of contacts are pre-defined: Administrative and Technical. You can add more contact types to the list using the wm.tn.dictionary:addContactType service.
External ID Types	A standard set of external ID types (for example, DUNS, DUNS+4, and User Defined 1) are provided. You can define more external ID types to the list using the wm.tn.dictionary:addIDType service.

Summary of Elements in this Folder

The elements that are available in this folder are listed in the following table:

Element	Description
wm.tn.dictionary:addContactType	Adds a contact type.
wm.tn.dictionary:addFieldDefinition	Adds a definition for a new extended profile field.

Element	Description
wm.tn.dictionary:addFieldGroup	Adds a new field group to the trading network. After you add a new field group, you can add extended field definitions associated with the new field group.
wm.tn.dictionary:addIDType	Adds an external ID type.
wm.tn.dictionary:addProfileGroup	Adds a new profile group to the trading network. After you add a new profile group, you can add partner profiles associated with the new profile group.
wm.tn.dictionary:changeContactType	Updates the description of a contact type.
wm.tn.dictionary:changeFieldGroup	Updates the name of a field group.
wm.tn.dictionary:changeIDType	Updates the description of an external ID type.
wm.tn.dictionary:changeProfileGroup	Updates the name of a profile group.
wm.tn.dictionary:deleteContactType	Deletes a contact type from Trading Networks.
wm.tn.dictionary:deleteFieldGroup	Deletes an extended field group from Trading Networks.
wm.tn.dictionary:deleteIDType	Deletes an external ID type from Trading Networks.
wm.tn.dictionary:deleteProfileGroup	Deletes a profile group fromTrading Networks.
wm.tn.dictionary:getBinaryTypes	Retrieves the binary types defined for the trading network.
wm.tn.dictionary:getContactTypes	Retrieves the contact types defined for the trading network.
wm.tn.dictionary:getFieldDefinitions	Retrieves profile field definitions. A profile field definition describes several aspects of a field-the field's name, data type, description, maximum length, default value, valid values, whether it is required, whether it is enabled or disabled, and to which field group it belongs.
wm.tn.dictionary:getFieldGroups	Retrieves the field groups defined for the trading network.
wm.tn.dictionary:getIDTypes	Retrieves all the external ID types defined for the trading network.
wm.tn.dictionary:getFieldGroups	Retrieves the profile groups defined for the trading network.
wm.tn.dictionary:updateFieldDefinition	Updates the definition for an existing standard or extended profile field.

wm.tn.dictionary:addContactType

Adds a contact type.

Input Parameters

description	**String** The new contact type to be added.

Output Parameters

id	**String** The ID that Trading Networks assigns to the contact type.

Usage Notes

After you add a contact type, you can add contacts for partners using the new contact type. To do this, set the *TypeID* for the contact to the ID that this service returns.

wm.tn.dictionary:addFieldDefinition

Adds a definition for a new extended profile field.

Input Parameters

definition	**Object** The new field definition to be added. The field definition must be a ProfileFieldMetaData object.

Output Parameters

None.

wm.tn.dictionary:addFieldGroup

Adds a new field group to the trading network. After you add a new field group, you can add extended field definitions that are associated with the new field group.

Input Parameters

description	**String** The name of the new field group to be added.

Output Parameters

id	**String** The ID that Trading Networks assigns to the field group.

Usage Notes

After you add a field group, you can associate extended field definitions to it. To make this association, set the *GroupID* in the field definition to the ID that this service returns.

wm.tn.dictionary:addIDType

Adds an external ID type.

Input Parameters

description	**String** The new external ID type to be added.

Output Parameters

id	**String** The internal ID that Trading Networks assigns to the external ID type.

Usage Notes

After you add an external ID type, you can specify an external ID of this type in a partner's profile. Set the *IDKey* for the external ID to the ID that this service returns.

wm.tn.dictionary:addProfileGroup

Adds a new profile group to the trading network. After you add a new profile group, you can add partner profiles associated with the new profile group.

Input Parameters

profileGroupName	**String** The name of the new profile group to be added.

Output Parameters

profileGroupId	**String** The ID that Trading Networks assigns to the profile group.

Usage Notes

After you add a new profile group, you can add partner profiles to this profile group. To make this association, set the *profileGroupIds* in the service to the ID that this service returns.

wm.tn.dictionary:changeContactType

Updates the description of a contact type.

Input Parameters

id	**String** The ID of the contact type to be updated.
description	**String** The new description of the contact type.

Output Parameters

None.

Usage Notes

You can only change the description of a contact type; you cannot change a Contact Type's ID.

wm.tn.dictionary:changeFieldGroup

Updates the name of a field group.

Input Parameters

id	**String** The ID of the field group to be updated.
description	**String** The new name for the field group.

Output Parameters

None.

Usage Notes

You can only change a field group's name; you cannot change a field group's ID.

wm.tn.dictionary:changeIDType

Updating the description of an external ID type.

Input Parameters

id	**String** The ID of the external ID type to be updated.

description	**String** The new description of the external ID type.

Output Parameters

None.

Usage Notes

You can only change the external ID type's description; you cannot change an external ID type's ID.

wm.tn.dictionary:changeProfileGroup

Updates the name of a profile group.

Input Parameters

profileGroupId	**String** The ID of the profile group to be updated.
profileGroupName	**String** The new name for the profile group.

Output Parameters

None.

Usage Notes

You can only change a profile group's name; you cannot change a profile group's ID.

wm.tn.dictionary:deleteContactType

Deletes a contact type from Trading Networks.

Input Parameters

id	**String** The ID of the contact type to be deleted.

Output Parameters

None.

Usage Notes

You cannot delete a contact type if any profile has a contact of the type you want to delete. If you attempt this, the service throws a Service Exception.

wm.tn.dictionary:deleteFieldGroup

Deletes an extended field group from Trading Networks.

Input Parameters

id	**String** The ID of the field group to be deleted.

Output Parameters

None.

Usage Notes

You cannot delete a field group if existing profile field definitions are associated with the field group. If you attempt this, the service throws a Service Exception.

wm.tn.dictionary:deleteIDType

Deletes an external ID type from Trading Networks.

Input Parameters

id	**String** The ID of the external ID type to be deleted.

Output Parameters

None.

Usage Notes

You cannot delete an external ID type if any partner's profile has an external ID of this type. If you attempt this, the service throws a Service Exception.

wm.tn.dictionary:deleteProfileGroup

Deletes a profile group from Trading Networks.

Input Parameters

profileGroupId	**String** The ID of the profile group to be deleted.

Output Parameters

None.

Usage Notes

You cannot delete a profile group if existing partner profiles are associated with the profile group. If you attempt this, the service throws a Service Exception.

wm.tn.dictionary:getBinaryTypes

Retrieves the binary types defined for the trading network.

Input Parameters

None.

Output Parameters

binaryTypes	**Object** A java.lang.Hashtable that contains the binary types. The Hashtable is keyed by the *String* binary type descriptions, and the values are the *Integer* ID type codes.
binaryTypesByID	**Object** A java.lang.Hashtable that contains the binary types. The Hashtable is keyed by the *Integer* ID type codes and the values are the *String* binary type descriptions.
descriptions	**String List** A sorted list of all binary type descriptions, returned as a string[]. The list is sorted in ascending alphabetical sequence.

Usage Notes

The three output variables contain the same data presented three different ways.

1. If you have the binary type description and you need its ID, use the Hashtable returned in *binaryTypes*.
2. If you have the binary type ID and you need its description, use the Hashtable returned in *binaryTypesByID*.
3. To present the binary types sorted in alphabetical order, use the list returned in *descriptions*. Then use the Hashtable in *binaryTypes* to look up the corresponding ID for any String in the *descriptions* list.

wm.tn.dictionary:getContactTypes

Retrieves the contact types defined for the trading network.

Input Parameters

None.

Output Parameters

contactTypes	**Object** A java.lang.Hashtable that contains the contact types. The Hashtable is keyed by the *String* contact type descriptions, and the values are the *Short* contact type codes.
contactTypesByID	**Object** A java.lang.Hashtable that contains the contact types. The Hashtable is keyed by the *Short* contact type codes and the values are the *String* contact type descriptions.
descriptions	**String List** A sorted list of all contact type descriptions, returned as a string[]. The list is sorted in ascending alphabetical sequence.

Usage Notes

The three output variables contain the same data presented three different ways.

1. If you have the contact type description and you need its ID, use the Hashtable returned in *contactTypes*.
2. If you have the contact type ID and you need its description, use the Hashtable returned in *contactTypesByID*.
3. To present the contact types sorted in alphabetical order, use the list returned in *descriptions*. Then use the Hashtable in *contactTypes* to look up the corresponding ID for any String in the *descriptions* list.

wm.tn.dictionary:getFieldDefinitions

Retrieves profile field definitions. A profile field definition describes several aspects of a field-the field's name, data type, description, maximum length, default value, valid values, whether it is required, whether it is enabled or disabled, and to which field group it belongs.

Input Parameters

type	**String** (optional) Indicates whether you want to retrieve standard or extended profile field definitions. The following values apply: ■ `standard` - Retrieves standard profile field definitions.

- `extended` - Retrieves extended profile field definitions.
- `null` - Retrieves both standard and extended profile field definitions

Output Parameters

definitions	**Document list** All the field definitions of the type specified: standard; extended or both. For flow programmers, each IS document (IData object) in *definitions* is represented by wm.tn.rec:FieldMetaData. For Java programmers, each field definition is a ProfileFieldMetaData object.

wm.tn.dictionary:getFieldGroups

Retrieves the field groups defined for the trading network.

Input Parameters

None.

Output Parameters

groups	**Object** A java.lang.Hashtable that contains the field groups. The Hashtable is keyed by the *String* field group descriptions, and the values are the *Short* field group codes.
groupsByID	**Object** A java.lang.Hashtable that contains the field groups. The Hashtable is keyed by the *Short* field group codes and the values are the *String* field group descriptions.
descriptions	**String List** A sorted list of all field group descriptions, returned as a String[]. The list is sorted in ascending alphabetical sequence.

Usage Notes

The three output variables contain the same data presented three different ways.

1. If you have the field group description and you need its ID, use the Hashtable returned in *groups*.
2. If you have the field group ID and you need its description, use the Hashtable returned in *groupsByID*.
3. To present the groups sorted in alphabetical order, use the list returned in *descriptions*. Then use the Hashtable in *groups* to look up the corresponding ID for any String in the *descriptions* list.

wm.tn.dictionary:getIDTypes

Retrieves all the external ID types defined for the trading network.

Input Parameters

None.

Output Parameters

idTypes	**Object** A java.lang.Hashtable that contains the external ID types. The Hashtable is keyed by the string external ID type descriptions, and the values are the short external ID type codes.
idTypesByID	**Object** A java.lang.Hashtable that contains the external ID types. The Hashtable is keyed by the short external ID types codes and the values are the string external ID types descriptions.
descriptions	**String List** A sorted list of all external ID type descriptions, returned as a string[]. The list is sorted in ascending alphabetical sequence.

Usage Notes

The three output variables contain the same data presented three different ways.

1. If you have the external ID type description and you need its ID, use the Hashtable returned in *idTypes*.
2. If you have the external ID type ID and you need its description, use the Hashtable returned in *idTypesByID*.
3. To present the external ID types sorted in alphabetical order, use the String List returned in *descriptions*. Use the Hashtable in *idTypes* to look up the corresponding ID for any String in the *descriptions* list.

wm.tn.dictionary:getProfileGroups

Retrieves the profile groups defined for the trading network.

Input Parameters

None.

Output Parameters

profileGroups	**Object** A java.lang.Hashtable that contains the profile groups. The Hashtable is keyed by the *String* profile group names, and the values are the *String* profile group IDs.
profileGroupIDs	**Object** A java.lang.Hashtable that contains the profile groups. The Hashtable is keyed by the *String* profile group IDs and the values are the *String* profile group names.

Usage Notes

The three output variables contain the same data presented three different ways.

1. If you have the profile group description and you need its ID, use the Hashtable returned in *profileGroups*.
2. If you have the profile group ID and you need its name, use the Hashtable returned in *profileGroupIDs*.

wm.tn.dictionary:updateFieldDefinition

Updates the definition for an existing standard or extended profile field.

Input Parameters

definition	**Object** The field definition to be updated. The field definition must be a ProfileFieldMetaData object.

Output Parameters

None.

Usage Notes

For standard profile fields, you can only update the field's description and whether the field is required. For extended profile fields, you can update any data except the internal field ID. Do *not* use `ProfileFieldMetaData.setFieldID` to change this value.

Example

To disable an extended profile field, perform the following:

1. Invoke wm.tn.dictionary:getFieldDefinitions specifying `extended` for *type*.

2. Select the extended profile field that you want to disable from the returned array of ProfileFieldMetaData objects.
3. Using the Java API, call the delete method on the ProfileFieldMetaData object.
4. Save the changes to the field definition by invoking this service.

7 Doc Folder

Overview

Use document services (services in the wm.tn.doc folder) to:

- View and manipulate business documents
- Validate the structure of flat file documents and to change the content types that are handled by the flat file content handler
- Manipulate XML business documents in Trading Networks

Summary of Elements in this Folder

- **Document Services.** Use document services (services in thewm.tn.doc folder) to view and manipulate business documents.

The elements that are available in this folder are listed in the following table:

Element	Description
wm.tn.doc:addAttributes	Adds custom attributes to a document.
wm.tn.doc:addContentPart	Adds a new content part to a document. A content part can be, for example, a segment of a document or an attachment.
wm.tn.doc:changeStatus	Changes the user status for a document.
wm.tn.doc:createNewEnvelope	Creates a new BizDocEnvelope that contains no document content.
wm.tn.doc:createReply	Creates a reply document for a specified document.
wm.tn.doc:deleteDocuments	Deletes documents that meet the specified criterion from the database. In addition to deleting documents, this service deletes the associated attributes, activity log entries, delivery tasks and relationships to and from this document.
wm.tn.doc:getContentPart	Retrieves a content part from the specified document. A content part can be, for example, a segment of a document or an attachment.
wm.tn.doc:getContentPartData	Retrieves the content of a content part.
wm.tn.doc:getDeliveryContent	Retrieves the delivery content of the specified document.
wm.tn.doc:getEvents	Retrieves the activity log entries (events) that are associated with a specified document.
wm.tn.doc:getSenderReceiver	Retrieves the sender and receiver information for the specified document from the Trading Networks database.

Element	Description
wm.tn.doc:handleLargeDoc	Submits an inputStream to Trading Networks through a content handler so that the pipeline is formatted as if the inputStream had been submitted by an external client.
wm.tn.doc:persist	Saves the supplied document to the Trading Networks database.
wm.tn.doc:recognize	Receives a document that Trading Networks is to recognize and returns a BizDocEnvelope that Trading Networks recognizes based on the defined set of document types. For flat files, this service also returns an IS document (IData object, TN_parms) that holds "hints" that Trading Networks uses for flat file document recognition.
wm.tn.doc:relateDocuments	Creates a one-way relationship between two documents.
wm.tn.doc:replaceContentPart	Replaces an existing content part of a document with the supplied content part. A content part can be, for example, a segment of a document or an attachment.
wm.tn.doc:resubmit	Extracts the document content from a BizDocEnvelope in the database and resubmits the document content to Trading Networks to be processed as a new document.
wm.tn.doc:resubmits	Extracts the document content from one or more BizDocEnvelopes in the database and resubmits the content of the documents to Trading Networks to be processed as a new documents.
wm.tn.doc:setAttribute	Updates, deletes, or adds an attribute value for a document in the BizDocEnvelope in memory.
wm.tn.doc:sign	Invokes the document verification service associated with the specified document to generate a digital signature for the document.
wm.tn.doc:updateAttributes	Updates custom attributes of a document in the database.
wm.tn.doc:updateComments	Updates the comment associated with a document in the database.
wm.tn.doc:updateSystemAttributes	Updates system attributes of a document.
wm.tn.doc:validate	Invokes the document validation service associated with the specified document to validate the structure of the document.

Element	Description
wm.tn.doc:verify	Invokes the document verification service associated with the specified document to verify the digital signature on the document.
wm.tn.doc:view	Retrieves a single document (envelope information and attributes) from the database; the service verifies that the client invoking the service is either the sending or receiving partner of the document being viewed or a Trading Networks administrator.
wm.tn.doc.viewAll	Retrieves a list of documents (envelope information) from the database; the service verifies that the client invoking the service is either the sending or receiving partner of the document being viewed or a Trading Networks administrator. Optionally, you can have the service retrieve the raw document contents, attributes and errors.
wm.tn.doc:viewAs	Retrieves a single document (envelope information and attributes) from the database; this service does not require the client invoking the service be a sender or receiver of the document being viewed.

- **Flat File Document Services.** Use flat file document services (services in the wm.tn.doc.ff folder) to validate the structure of a flat file document and to change which content types will be handled by the flat file content handler.

The elements that are available in this folder are listed in the following table:

Element	Description
wm.tn.doc.ff:registerContentTypes	Trading Networks uses this service to register those content types that will be handled by the flat file content handler.
wm.tn.doc.ff:routeFlatFile	Recognizes a flat file document and submits it for processing.
wm.tn.doc.ff:validate	Validates the structure of a flat file document.

- **XML Document Services.** Use XML document services to manipulate XML business documents in Trading Networks.

The elements that are available in this folder are listed in the following table:

Element	Description
wm.tn.doc.xml:bizdocToRecord	Transforms a business document into an IS document (IData object) based on the IS document type blueprint associated with the TN document type (if any).

Element	Description
wm.tn.doc.xml:recordToBizdoc	Transforms an IS document (IData object) into an XML document and sends the resulting XML document into the document recognition engine to translate the XML document into a business document.
wm.tn.doc.xml:routeXml	Recognizes an XML document and submits it for processing.

wm.tn.doc:addAttributes

Adds custom attributes to a document.

Input Parameters

bizdoc **Object** The document to which you want to add custom attributes. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.

attrNames **String List** The list of custom attributes to add to the document.

Overwrite **String** Indicates whether the custom attributes from the supplied data should replace the custom attributes in the database. Valid values are:

- `true` - Replaces the custom attributes in the database with those in data.
- `false` - Appends the custom attributes in data to the custom attributes in the database.

Output Parameters

addCount **String** The number of custom attributes that the service added.

wm.tn.doc:addContentPart

Adds a new content part to a document. A content part can be, for example, a segment of a document or an attachment.

Input Parameters

bizdoc **Object** The document to which you want to add a new content part. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.

partName **String** The name of the new content part (for example, “OrderAttachment”).

partBytes	**Object** The content of the part you are adding to the document. The data type of *partBytes* must be byte[].
partStream	**Object** The content of the part you are adding to the document. The data type of *partStream* must be java.io.InputStream.
mimeType	**String** The MIME type of the content part you are adding (e.g. "text/plain" or "application/pdf").
partIndex	**String** (optional) The position of the content part in the document's existing array of parts. The beginning position of the content part is 0.

Output Parameters

updateCount	**String** The number of documents that the service updated. The values indicate: ■ `1` The service added the new content part to the document. ■ `0` The service did not add the new content part to the document.

Usage Notes

This service updates the document in memory. If the document has been saved, the service also updates the database.

You must supply either *partBytes* or *partStream* to the addContentPart service, but not both. If *partStream* is supplied, Trading Networks will determine whether the new content part is large, and will handle it appropriately. If *partBytes* is supplied, Trading Networks always considers the new content part to be small, regardless of its actual size. For information about large document handling, see *webMethods Trading Networks Administrator's Guide*.

wm.tn.doc:changeStatus

Changes the user status for a document.

Input Parameters

bizdoc	**Object** The document for which you want to change the status. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
docID	**String** The internal document ID of the document.
newUserStatus	**String** (optional) The new user status for the document.

Output Parameters

updateCount	**String** The number of documents that the service updated. The following values indicate: - `1` - The service changed the status of the document. - `0` - The service did not change the status of the document.

Usage Notes

- This service updates the version of the document in memory and the version in the database if the document was saved.
- An alternative to wm.tn.doc:changeStatus is the wm.tn.doc:updateSystemAttributes service. You can use the updateSystemAttributes service to change the user status. If the pre-processing actions indicate that the document attributes are to be saved to the database, the updateSystemAttributes service writes the changes to the database and creates a detailed record of the change in the Trading Networks Activity Log.
- In earlier version of Trading Networks, this service provided the ability to modify the document's processing (system) status. The process (system) status is reserved for internal use only by Trading Networks and should *not* be modified by an application. This service has been change to only allow for modification of the user status.

wm.tn.doc:createNewEnvelope

Creates a new BizDocEnvelope that contains no document content.

Input Parameters

typeId	**String** The internal ID for the type of the document you want to create. For *typeId*, specify the unique identifier that Trading Networks generated for the TN document type. This is the value returned by the getId method of the BizDocType class.
senderId	**String** (optional) The internal identifier for the sender of the new document. Trading Networks assigns this unique ID to the partner when the profile for the partner was added to your network.
receiverId	**String** (optional) The internal identifier for the receiver of the new document. Trading Networks assigns this unique ID to the partner when the profile for the partner was added to your network.
documentId	**String** (optional) The document ID of the document. This is an identifier for the document in some external identifying scheme.

groupId	**String** (optional) An identifier for the group to which this new document belongs.
conversationId	**String** (optional) An identifier for the conversation to which this new document belongs.

Output Parameters

bizdoc	**Document** The new bizdoc envelope. For the structure of *bizdoc*, see wm.tn.rec:BizDocEnvelope.

Usage Notes

If you are invoking this service from a Java program, in addition to returning *bizdoc* as an IS document (IData object), the service returns *bizdoc* as an instance of com.wm.app.tn.doc.BizDocEnvelope.

wm.tn.doc:createReply

Creates a reply document for a specified document.

Input Parameters

bizdoc	**Object** The document for which you want to create a reply document. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
replyTypeId	**String** (optional) The internal ID for the TN document type to associate with the reply document (if different from the original document). For *replyTypeId*, specify the unique identifier that Trading Networks generated for the TN document type. This is the value from the TypeID column in the BizDocTypeDef table in the Trading Networks database.
replyDocRelationship	**String** (optional) Type of relationship you want to create between the original document and the reply document. You can specify a string between 1-80 characters. A relationship is created *only* if you specify *replyDocRelationship*.

Output Parameters

reply	**Object** The reply document. For the structure of *reply*, see wm.tn.rec:BizDocEnvelope.

Usage Notes

If you are invoking this service from a Java program, in addition to returning *reply* as an IS document (IData object), the service returns *reply* as an instance of com.wm.app.tn.doc.BizDocEnvelope.

wm.tn.doc:deleteDocuments

Deletes documents that meet the specified criterion from the database. In addition to deleting documents, this service deletes the associated attributes, activity log entries, delivery tasks and relationships to and from this document.

Optionally, you can select to have this service delete documents that are related to the documents being deleted.

Input Parameters

docId	**String** (optional) The internal document ID of the document to delete.
systemStatus	**String** (optional) The system status of the document(s) to delete.
userStatus	**String** (optional) The user status of the document(s) to delete.
docTypeName	**String** (optional) The name of the TN document type that was used for the document(s) to delete.
deleteRelated	**String** (optional) Indicates whether to delete documents that are related to the document(s) being deleted. Valid values are: - `true` - Delete the document(s) that match the specified criteria and the related document(s). - `false` - Delete only the document(s) that match the specified criteria.

Output Parameters

updateCount	**String** Returns the number of documents that this service deleted from the database.

wm.tn.doc:getContentPart

Retrieves a content part from the specified document. A content part can be, for example, a segment of a document or an attachment.

Input Parameters

bizdoc	**Object** The document from which you want to retrieve a content part. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
partName	**String** The name of the content part you want to retrieve (for example, "xmldata").

Output Parameters

contentPart	**Document** The content part that was retrieved. For the structure of *contentPart*, see wm.tn.rec:BizDocContentPart.

wm.tn.doc:getContentPartData

Retrieves the content of a content part. You can use this service to get content of small, as well as large content parts. It allows you to get the content as an InputStream or as byte[]. If you request the content to be returned in byte[] format, you must specify the start index and the number of bytes you need.

Input Parameters

bizdoc	**Object** The document from which you want to get content. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
partName	**String** The name of the content part from which you want get content.
getAs	**String** Specifies the format in which the content should be returned. Set the following values: ■ `bytes` - Returns content as a byte[] object. If you specify `bytes`, you must specify *startIndex* and *byteCount*. ■ `stream` - Returns an *InputStream* from the content of this part.
startIndex	**String** (optional) If *getAs* is set to `bytes`, this specifies the starting index of the content from which to read. This input is optional when *getAs* is set to `stream`.
byteCount	**String** (optional) If *getAs* is set to *bytes*, this specifies the number of bytes to read from *startIndex*. This input is optional when *getAs* is set to `stream`.

Output Parameters

partContent	**Object** The content of the part specified by *partName*. If *getAs* is set to `bytes`, *partContent* is an instance of byte[]. If *getAs* is set to `stream`, *partContent* is an instance of *InputStream*.

wm.tn.doc:getDeliveryContent

Retrieves the delivery content of the specified document.

In addition to returning delivery content, this service returns the content type and ftp file extension of the given document.

Input Parameters

bizdoc	**Object** The document for which to get the delivery content. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.

Output Parameters

deliveryContent	**Object** The delivery content of the specified document. *deliveryContent* can be in one of the two formats byte[] (mostly for small documents) and InputStream (mostly for large documents).
content-type	**String** Returns the content type of this document, which can be used in transport headers.
ftpFileExtension	**String** Returns the file extension that can used when naming files that are sent through ftp transport.

wm.tn.doc:getEvents

Retrieves the activity log entries (events) that are associated with a specified document.

Input Parameters

internalId	**String** The internal document ID of the document for which to retrieve activity log entries. This is a unique ID that Trading Networks assigns to the document.
errorsOnly	**String** (optional) Whether you want the service to retrieve all entries or only error entries. Valid values are: ■ `true` - Retrieves only error entries.

- `false` - Retrieves all entries.

Output Parameters

eventCount	**String** The number of activity log entries that the service returned.
errorCount	**String** The number of activity log entries of type "Error" that the service returned. If you specified `true` for *errorsOnly*, the value of *errorCount* will be equal to *eventCount*.
events	**Document List** A list of activity log entries associated with the document. For the structure of each activity log entry that is returned in the *events* document list, see wm.tn.rec:ActivityLogEntry.

Usage Notes

The behavior of this service has been changed to return the date and time in the long format only if the value of the isMWS parameter while invoking the wm.tn.query:doQuery service is set to "yes". In all other cases, this service returns the date and time in the yyyy-MM-dd HH:mm:ss format.

wm.tn.doc:getSenderReceiver

Retrieves the sender and receiver information for the specified document from the Trading Networks database.

If the pipeline already contains *sender*, a ProfileSummary, only the receiver information will be retrieved.

Input Parameters

bizdoc	**Object** The document for which you want to retrieve sender and receiver information. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.

Output Parameters

sender	**Document** The profile summary for the sender of the document. For the structure of *sender*, see wm.tn.rec:ProfileSummary.
receiver	**Document** The profile summary for the receiver of the document. For the structure of *receiver*, see wm.tn.rec:ProfileSummary.

Usage Notes

If you are invoking this service from a Java program, in addition to returning *sender* and *receiver* as IS documents (IData objects), the service returns *sender* and *receiver* as an instances of com.wm.app.tn.profile.ProfileSummary.

wm.tn.doc:handleLargeDoc

Submits an inputStream to Trading Networks through a content handler so that the pipeline is formatted as if the inputStream had been submitted by an external client.

Input Parameters

inputStream	**Object** The inputStream to submit to Trading Networks.
content-type	**String** The content-type of the *inputStream*. The value can be a standard content type, such as text/xml or image/gif, or a custom type. The content-type should be a content type for which a content handler has been registered with Integration Server.
content-length	**String** The length of *inputStream*.

Output Parameters

None.

Usage Notes

- The output of the wm.tn.doc:handleLargeDoc service varies depending on the value of the following:
  - The *content-type* variable
  - The Content Handler registered to handle that type of content
  - The length of the content specified in the *content-length* variable

    For example, if the *content-type* is text/xml and no custom content handler has been registered for that type, the built-in Trading Networks XML content handler is used to format the pipeline. In this case, if the length is less than the value of the tn.BigDocThreshold system property, the pipeline will contain a variable of type com.wm.lang.xml.Document, named node. If the length is greater than or equal to tn.BigDocThreshold, the pipeline will contain a variable of type com.wm.util.tspace.Reservation, named *$reservation*.
- Any items that are in the pipeline when the wm.tn.doc:handleLargeDoc service is invoked will also be included in the pipeline that is produced by this service.
- This service is useful when you have a facility running within the Integration Server that needs to send a document into Trading Networks. Without this service, you would use the pub.client:http service to create a connection to Integration Server and transmit the document that way. It is

much more efficient to invoke this service to format the pipeline, invoke wm.tn:receive, wm.tn.doc:routeXml, or wm.tn.doc:routeFlatFile.

wm.tn.doc:persist

Saves the supplied document to the Trading Networks database.

Input Parameters

bizdoc	**Object** The document you want to save to the database. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
flags	**Object** (optional) Flags that specify the pre-processing actions for the document. If specified, the service uses the *persist?* flag to determine whether to save the document. The flags must be an instance of com.wm.app.tn.route.PreRoutingFlags.

Output Parameters

None.

Usage Notes

- This service saves the document *only* if it has not already been saved to the database.
- If *flags* is non-null, the service uses the *persist?* flag to determine whether to save the document. For the format of *flags*, see wm.tn.rec:PreProcessingFlags.

wm.tn.doc:recognize

Receives a document that Trading Networks is to recognize and returns a BizDocEnvelope that Trading Networks recognizes based on the defined set of document types. For flat files, this service also returns an IS document (IData object, TN_parms) that holds "hints" that Trading Networks uses for flat file document recognition.

Input Parameters

TN_parms	**Document** (Optional) An IS document (IData object) that you can use to provide parameters that govern how Trading Networks recognizes and processes a document. *TN_parms* is primarily used for flat file processing. The document gateway service adds "hints" to *TN_parms* that Trading Networks uses when performing document recognition for a flat file document. See information about document gateway services in *webMethods Trading Networks Administrator's Guide* for details on providing recognition hints.

For both XML and flat files, you can optionally add the *TN_parms/DoctypeID* or *TN_parms/DoctypeName* fields.

The *TN_parms/DoctypeID* and *TN_parms/DoctypeName* fields identify the TN document type to use, thus bypassing document recognition and eliminating the overhead of searching for the TN document type. If you specify both variables, *DoctypeID* is used.

- *TN_parms/DoctypeID* is a string that identifies the internal identifier of the TN document type. To determine the identifier use the wm.tn.doctype:list service. Using *DoctypeID* rather than *DoctypeName* is more stable because the internal identifier cannot be changed.
- *TN_parms/DoctypeName* is a string that identifies the name of the TN document type. Be sure to use the exact combination of upper- and lowercase letters.

Note:
You can add a flat file document as an Object with the name ffdata in the pipeline

For XML documents you can optionally add a node (**Object**). The document to process must be an instance of com.wm.lang.xml.Document. The typical way to get an XML document into the pipeline is by posting an XML document to Integration Server.

For EDI documents, you can add an EDI document as an Object with the name edidata in the pipeline.

Output Parameters

bizdoc	**Document** (optional) The document that Trading Networks received (that is, the document passed in the node input variable) formatted as an IS document (IData object). For the structure of bizdoc, see wm.tn.rec:BizDocEnvelope.
TN_parms	**Document** (optional) An IS document (IData object) that holds "hints" that Trading Networks uses when performing document recognition for a flat file document. For information about document gateway services and recognition hints, see *webMethods Trading Networks Administrator's Guide*.

wm.tn.doc:relateDocuments

Creates a one-way relationship between two documents.

Input Parameters

fromDoc	**Object** The document the relationship is "from." The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
toDoc	**Object** The document the relationship is "to." The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
relationship	**String** A string that describes the type of relationship between the documents. The string can be 1-80 characters.

Output Parameters

updateCount	**String** The number or documents that the service updated. The following values indicate: ■ `1` The service established the relationship. ■ `0` The service did not establish the relationship.

Usage Notes

One of the documents must have been saved to the database before invoking this service. If neither have been saved to the database, this service throws an exception.

wm.tn.doc:replaceContentPart

Replaces an existing content part of a document with the supplied content part. A content part can be, for example, a segment of a document or an attachment.

Input Parameters

bizdoc	**Object** The document in which you want to replace a content part. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
partName	**String** (optional) The name of the content part (for example, "OrderAttachment") that must be replaced.
partBytes	**Object** The content of the part you are replacing in the document. The data type of *partBytes* must be byte[].
partStream	**Object** The content of the part you are replacing in the document. The data type of *partStream* must be java.io.InputStream.
mimeType	**String** The MIME type of the content part you are replacing (for example, "text/plain" or "application/pdf".)

partIndex	**String** (optional) The position of the content part in the document's existing array of parts.

Output Parameters

updateCount	**String** The number of documents that the service updated. Valid values are: ■ `1` The service replaced the content part of the document successfully. ■ `0` The service did not replace any content part of the document.

Usage Notes

This service replaces the content part of the saved document and also updates the database.

If you supply only the *partName*, the *partIndex* is retrieved from the database. Also, if you supply only the *partIndex*, the *partName* is retrieved from the database.

You must supply either *partBytes* or *partStream* to the replaceContentPart service, but not both. If *partStream* is supplied, Trading Networks will determine whether the new content part is large, and will handle it appropriately. If *partBytes* is supplied, Trading Networks always considers the new content part to be small, regardless of its actual size. For information about large doc handling, see *webMethods Trading Networks Administrator's Guide*.

wm.tn.doc:resubmit

Extracts the document content from a BizDocEnvelope in the database and resubmits the document content to Trading Networks to be processed as a new document.

To process the document Trading Networks invokes the same receive service that the original document used, providing the *ReceiveSvc* field was set on the original BizDocEnvelope. The receive service is the service the original document was sent to for processing, for example, a document gateway service.

For Trading Networks to determine the service to use for resubmitting a flat file document, the document gateway service must populate the variable in the pipeline, *TN_parms/$receiveSvc*, with its service name. Trading Networks uses the value of *TN_parms/$receiveSvc* to set the *ReceiveSvc* field of the BizDocEnvelope. For information about flat file document types, see *webMethods Trading Networks Administrator's Guide*. For information about resubmitting flat file documents, see *webMethods Trading Networks User's Guide*.

Input Parameters

internalId	**String** The Trading Networks-generated internal ID of the document to resubmit.

relationship	**String** (optional) A string that describes the type of relationship to create between the original document and the resubmitted document. The string can be 1-80 characters. The default relationship is "RESUBMIT".

Output Parameters

bizdoc	**Object** (optional) The new BizDocEnvelope that Trading Networks created for the document content that was resubmitted. It is an instance of com.wm.app.tn.doc.BizDocEnvelope.
originalDoc	**Object** (optional) The BizDocEnvelope for the original document content that was resubmitted and that was identified by the *internalID* input variable. It is an instance of com.wm.app.tn.doc.BizDocEnvelope.
$tnReprocess	**String** (optional) An internal variable that Trading Networks uses to distinguish a resubmission from an original submission.
TN_parms	**Document** (optional) An IData object (IS document) holding internal data that Trading Networks uses.

Usage Notes

If the `tn.resubmit.return.bizdocs` system property is `true`, this service returns as output both the *bizdoc* and *originalDoc*. The default for the `tn.resubmit.return.bizdocs` property is `false`, which indicates that the wm.tn.doc:resubmit service should return neither the *bizdoc* nor *originalDoc*.

wm.tn.doc:resubmits

Extracts the document content from one or more BizDocEnvelopes in the database and resubmits the content of the documents to Trading Networks to be processed as a new documents.

To process each document Trading Networks invokes the same "receive" service that each original document used, if the *ReceiveSvc* field was set on the original BizDocEnvelope for the document. The "receive" service is the service the original document was sent to for processing, for example, a document gateway service.

For Trading Networks to know to which service it should resubmit a flat file document, the document gateway service must place its name in the *TN_parms*/*$receiveSvc* variable in the pipeline. Trading Networks then uses the value of the *TN_parms*/*$receiveSvc* variable to set the *ReceiveSvc* field of the BizDocEnvelope. For more information, see information about flat file document types in *webMethods Trading Networks Administrator's Guide* and resubmitting flat file documents in *webMethods Trading Networks User's Guide*.

Input Parameters

internalIds	**String List** The Trading Networks-generated internal ID of the documents to resubmit.
relationship	**String** (optional) A string that describes the type of relationship to create between each original document and its corresponding resubmitted document. The string can be 1-80 characters. The default relationship is "RESUBMIT".

Output Parameters

bizdoc	**Object List** (optional) The new BizDocEnvelopes that Trading Networks created for each document content that was resubmitted. Each BizDocEnvelope in the Object List is an instance of com.wm.app.tn.doc.BizDocEnvelope.
originalDocs	**Object List** (optional) The BizDocEnvelopes for the original documents that were resubmitted and that were identified in the *internalIDs* input variable. Each BizDocEnvelope in the Object List is an instance of com.wm.app.tn.doc.BizDocEnvelope.
$tnReprocess	**String List** (optional) Internal variable that Trading Networks uses to distinguish a resubmission from an original submission.
TN_parms	**Document List** (optional) An array of IData objects (IS documents) holding internal data that Trading Networks uses.

Usage Notes

If the `tn.resubmit.return.bizdocs` system property is `true`, this service returns as output both the *bizdoc* and *originalDoc*. The default for the `tn.resubmit.return.bizdocs` property is `false`, which indicates that the wm.tn.doc:updateAttributes service should return neither the *bizdoc* nor *originalDoc*.

wm.tn.doc:setAttribute

Updates, deletes, or adds an attribute value for a document in the BizDocEnvelope in memory.

This service supports attributes of the following data types: String, StringList, Number, NumberList, DateTime, and DateTimeList.

Input Parameters

bizdoc	**Object** The document for which you want to update, delete, or add an attribute value. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.

attribId	**String** (optional) Internal ID of the attribute to be set.
attribName	**String** (optional) Name of the attribute to be set.
attribValue	**String** (optional) New value for the attribute. If you supply a null value, the service deletes any existing value for the attribute. If the attribute is a DATETIME, use the format: yyyy-mm-dd hh:mm:ss.

Output Parameters

None.

Usage Notes

This service does *not* update the value of the attribute of the BizDocEnvelope saved in the Trading Networks database. To update the database with the values of the attributes in a BizDocEnvelope, use wm.tn.doc:updateAttributes.

wm.tn.doc:sign

Invokes the document verification service associated with the specified document to generate a digital signature for the document.

Input Parameters

bizdoc	**Object** The document you want to digitally sign. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
flags	**Object** (optional) Flags that specify the pre-processing actions for the document. If specified, the service uses the *verify?* flag to determine whether to create the digital signature. The flags must be an instance of com.wm.app.tn.route.PreProcessingFlags.

Output Parameters

None.

Usage Notes

- Every TN document type is associated with a signing service. The wm.tn.doc:sign service looks up the appropriate signing service and executes it against the supplied document. The service attaches all resulting errors to the document and logs them to the activity log in the Trading Networks database (if the document is saved to the database). To retrieve the signing errors from the activity log, use the wm.tn.doc:getEvents service.
- If the document is an outbound XML document and the receiver's profile in the Trading Networks system contains a private key and digital certificate, the XML signing service

generates a PKCS#7 detached digital signature for the supplied document, base 64 encodes it, and inserts it into the document in the location specified by the Signature query. The service uses the SignedBody query to determine what portion of the XML content to sign.

- If *flags* is non-null, the service uses the *verify?* flag to determine whether to sign the document. For the format of *flags*, see wm.tn.rec:PreProcessingFlags.

wm.tn.doc:updateAttributes

Updates custom attributes of a document in the database.

Input Parameters

bizdoc	**Object** The document for which you want to update custom attribute values in the database. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.

Output Parameters

updateCount	**String** The number of attributes that the service updated.

wm.tn.doc:updateComments

Updates the comment associated with a document in the database.

Input Parameters

docID	**String** The internal document ID of the document that you want to update.
comments	**String** The comment you want to associate with the document.

Output Parameters

updateCount	**String** The number of comments that the service updated.

wm.tn.doc:updateSystemAttributes

Updates system attributes of a document.

Input Parameters

bizdoc	**Object** The document for which you want to update system attribute values in the database. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.

Output Parameters

None.

Usage Notes

- You can use this service to change any of the system attributes in the BizDocEnvelope: *SenderId*, *ReceiverId, DocumentId, GroupId, ConversationId, SystemStatus,* and *UserStatus*. Modify the values you want to change; then invoke this service. If the pre-processing actions indicate that the document attributes are to be saved to the database, the updateSystemAttributes service writes the changes to the database and creates a detailed record of the change in the Trading Networks Activity Log.
- *SenderId* and *ReceiverId* are Trading Networks-generated IDs for partner profiles. These values must match an existing partner profile ID. If you change either of these fields to an invalid ID value, the service will throw an exception.
- Trading Networks uses *SystemStatus* to control the processing of the BizDocEnvelope. This field is reserved for internal use only by Trading Networks and should *not* be modified by an application. If you attempt to set this value to null, an exception is thrown.

wm.tn.doc:validate

Invokes the document validation service associated with the specified document to validate the structure of the document.

Input Parameters

bizdoc	**Object** The document for which you want to validate the structure. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
flags	**Object** (optional) Flags that specify the pre-processing actions for the document. If specified, the service uses the *validate?* flag to determine whether to validate the structure of the document. The flags must be an instance of com.wm.app.tn.route.PreProcessingFlags.

Output Parameters

None.

Usage Notes

- Every TN document type is associated with a validation service. The wm.tn.doc:getEvents service looks up the appropriate validation service and executes it against the supplied document. All resulting validation errors are attached to the document and logged to the activity log in the Trading Networks database if the document is saved to the database. To retrieve the validation errors from the activity log, use wm.tn.doc:getEvents.
- If the document is an XML document, the XML validation service validates its content against the IS schema selected for use with documents of the corresponding BizDocType.
- If *flags* is non-null, the service uses the *validate?* flag to determine whether to validate the structure of the document. For the format of *flags*, see wm.tn.rec:PreProcessingFlags.

wm.tn.doc:verify

Invokes the document verification service associated with the specified document to verify the digital signature on the document.

Input Parameters

bizdoc	**Object** The document for which you want to verify the signature. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
flags	**Object** (optional) Flags that specify the pre-processing actions for the document. If specified, the service uses the *verify?* flag to determine whether to verify the digital signature. The flags must be an instance of com.wm.app.tn.route.PreProcessingFlags.

Output Parameters

None.

Usage Notes

- Every TN document type is associated with a verification service. The wm.tn.doc:getEvents service looks up the appropriate verification service and executes it against the supplied document. The service attaches all resulting verification errors to the document and logs them to the activity log in the Trading Networks database (if the document is saved to the database). To retrieve the verification errors from the activity log, use wm.tn.doc:getEvents.
- The XML verification service uses the Signature and SignedBody queries to extract those portions of the document. The signature is extracted and base-64 decoded. The signed body is extracted and converted to bytes in the UTF8 encoding. The resulting byte data must have been signed with the extracted signature. Additionally, the digital certificate used to sign the byte data must be the same as the one in the profile for the document's sender. If all of these conditions are met, verification succeeds.

- If *flags* is non-null, the service uses the *verify?* flag to determine whether to verify the digital signature of the document. For the format of *flags*, see wm.tn.rec:PreProcessingFlags.

wm.tn.doc:view

Retrieves a single document (envelope information and attributes) from the database; the service verifies that the client invoking the service is either the sending or receiving partner of the document being viewed or a Trading Networks administrator.

Optionally, the service can retrieve the raw document content and related document information.

Input Parameters

internalId **String** The internal document ID of the document to retrieve. This is a unique ID that Trading Networks assigns to the document.

getContent **String** (optional) Indicates whether to retrieve the document content and envelope information. Valid values are:

- `true` - Retrieve the document content.
- `false`- Default. Do *not* retrieve the document content.

contentPartCriteria **Document** (optional) The retrieval criteria for the content parts associated with this document. This is an instance of wm.tn.rec:BizDocEnvelope. If not specified, this service retrieves all content parts of the matching document. Specify a list of part names to include and/or a list of part names to exclude from the retrieved envelope. The document contains the following keys:

- *includeParts* **String List** (optional) The list of content part names to retrieve.
- *excludeParts* **String List** (optional) The list of content part names that should not be retrieved.

getRelated **String** (optional) Whether you want to retrieve information about the related documents (including grouped documents and those in the conversation). Valid values are:

- `true` - Retrieve the related document information.
- `false` - Default. Do *not* retrieve the related document information.

Output Parameters

bizdoc **Document** The document this service retrieved from the database. For the structure of *bizdoc*, see wm.tn.rec:BizDocEnvelope. The service fills in the *Content* field within *bizdoc* if you specified `true` for *getContent*.

sender	**Document** The profile summary for the sender of the document. For the structure of *sender*, see wm.tn.rec:ProfileSummary.
receiver	**Document** The profile summary for the receiver of the document. For the *receiver*, see wm.tn.rec:ProfileSummary.
relatedDocCount	**String** (optional) If you specified `true` for *getRelated*, this String contains the number of related documents that are associated with the retrieved document.
relatedDocs	**Document List** (optional) If you specified `true` for *getRelated*, this is information about documents related to the retrieved document. Each IS document (IData object) returned in *relatedDocs* will contain these keys: ■ *relationship* **String** The type of relationship. ■ *from* **String** The internal document ID of the "from" document. ■ *to* **String** The internal document ID of the "to" document.
groupedDocCount	**String** (optional) If you specified `true` for *getRelated*, the number of documents with the same group ID as the returned document.
groupedDocs	**String List** (optional) If you specified `true` for *getRelated*, a list of documents with the same group ID as the returned document.

Usage Notes

If you are invoking this service from a Java program, in addition to returning *bizdoc*, *sender*, and *receiver* as IS documents (IData objects), the service returns *bizdoc* as an instance of com.wm.app.tn.doc.BizDocEnvelope and *sender* and *receiver* as an instances of com.wm.app.tn.profile.ProfileSummary.

wm.tn.doc.viewAll

Retrieves a list of documents (envelope information) from the database; the service verifies that the client invoking the service is either the sending or receiving partner of the document being viewed or a Trading Networks administrator. Optionally, the service can retrieve the raw document contents, attributes, and errors.

Input Parameters

internalIds	**String List** The internal document IDs of the documents to retrieve. These are unique IDs that Trading Networks assigns to documents.
getContents	**String List** (optional) Retrieves the document contents and envelope information. Valid values are: ■ `true` - Retrieve the document contents.

- `false` - Default. Do *not* retrieve the document contents.

getAttributes **String** (optional) Retrieves the document attributes and envelope information. Valid values are:

- `true` - Retrieve the document attributes.
- `false` - Default. Do *not* retrieve the document attributes.

getErrors **String** (optional) Retrieves the document errors and envelope information. Valid values are:

- `true` - Retrieve the document errors.
- `false` - Default. Do *not* retrieve the document errors.

Output Parameters

bizdocs **Document List** Documents this service retrieved from the database. For the structure of *bizdoc*, see wm.tn.rec:BizDocEnvelope. The service populates the *Content* attribute within each *bizdoc* if *getContents*, *getAttributes*, and *getErrors* is set as `true`.

Usage Notes

If you are invoking this service from a Java program, in addition to returning *bizdocs* as IS documents (IData object), the service returns *bizdocs* as an instance of com.wm.app.tn.doc.BizDocEnvelope.

wm.tn.doc:viewAs

Retrieves a single document (envelope information and attributes) from the database; this service does not require the client invoking the service be a sender or receiver of the document being viewed. Optionally, the service can retrieve the raw document content and related document information.

This service is intended for use by other webMethods components, such as webMethods Module for EDI.

Input Parameters

internalId **String** The internal document ID of the document to retrieve. This is a unique ID that Trading Networks assigns to the document.

getContent **String** (optional) Whether you to retrieve the document content (in addition to the envelope information). Valid values are:

- `true` - Retrieve the document content.

- `false` - Default. Do *not* retrieve the document content.

getRelated **String** (optional) Retrieves information about the related documents, grouped documents, and documents in the conversation. Valid values are:

- `true` - Retrieve the related document information.
- `false`- Default. Do *not* retrieve the related document information.

contentPartCriteria **Document** (optional) The retrieval criteria for the content parts associated with this document. This is an instance of wm.tn.rec:BizDocEnvelope. If not specified, this service retrieves all content parts of the matching document. You can specify a list of part names to include and/or a list of part names to exclude from the retrieved envelope. The document contains the following keys:

- *includeParts* **String List** (optional) The list of content part names to retrieve.
- *excludeParts* **String List** (optional) The list of content part names that should not be retrieved.

Output Parameters

bizdoc **Document** The document retrieved from the database. For the structure of *bizdoc*, see wm.tn.rec:BizDocEnvelope. The service populates the *Content* field within *bizdoc* if *getContent* is set as `true`.

sender **Document** The profile summary for the sender of the document. For the structure of *sender*, see wm.tn.rec:ProfileSummary.

receiver **Document** The profile summary for the receiver of the document. For the *receiver*, see wm.tn.rec:ProfileSummary.

relatedDocCount **String** (optional) If *getRelated* is set as `true`, this is the number of related documents associated with the retrieved document.

relatedDocs **Document List** (optional) If *getRelated* is set as `true`, this is information about documents related to the retrieved document. Each IS document (IData object) returned in *relatedDocs* contains the following keys:

- *relationship* **String** The type of relationship.
- *from* **String** The internal document ID of the "from" document.
- *to* **String** The internal document ID of the "to" document.

groupedDocCount **String** (optional) If *getRelated* is set as `true`, the number of documents with the same group ID as the returned document.

groupedDocs	**String List** (optional) If *getRelated* is set as `true` a list of documents with the same group ID as the returned document.

Usage Notes

If you are invoking this service from a Java program, in addition to returning *bizdoc*, *sender*, and *receiver* as IS documents (IData objects), the service returns *bizdoc* as an instance of com.wm.app.tn.doc.BizDocEnvelope and *sender* and *receiver* as an instance of com.wm.app.tn.profile.ProfileSummary.

wm.tn.doc.ff:registerContentTypes

Trading Networks uses this service to register those content types that are handled by the flat file content handler. This service is invoked when the WmTN package is loaded.

Input Parameters

None.

Output Parameters

None.

Usage Notes

By default, the flat file content handler processes only data with a content type of "text/plain." To modify the set of content types that the flat file content handler processes:

1. First, change the value of the `tn.ff.contenttypes` property in the `<SERVER_HOME>/packages/WmTN/config/properties.cnf` file.
2. Next, reload the WmTN package or restart Integration Server

   For example, set the property as follows.

   ```
   tn.ff.contenttypes=text/special,application/x-my-app
   ```

   When the WmTN package is reloaded, the flat file content handler processes all incoming data with a content type of `text/special` or `application/x-my-app`.

   Note:
   If you change the value of the `tn.ff.contenttypes` property and invoke this service without reloading WmTN, the change will not take effect.

wm.tn.doc.ff:routeFlatFile

Recognizes a flat file document and submits it for processing. This service does not validate the identity of the sender to the currently logged in user. Only invoke this service from within

processing rules or services; do not expose directly to trading partners. Trading partners should use the document gateway service.

Input Parameters

ffdata	**Object** The flat file document. For Java developers, this is an instance of java.io.InputStream.
TN_parms	**Document** (optional) An IS document (IData object) that holds "hints" that Trading Networks uses when performing document recognition for a flat file document. See information about document gateway services in *webMethods Trading Networks Administrator's Guide* for details on providing recognition hints.

Output Parameters

bizdoc	**Document** The flat file document formatted as an IS document (IData object). For the structure of *bizdoc*, see wm.tn.rec:BizDocEnvelope.
sender	**Document** The profile summary for the sender of the flat file document. For the structure of *sender*, see wm.tn.rec:ProfileSummary.
receiver	**Document** The profile summary for the receiver of the flat file document. For the structure of *receiver*, see wm.tn.rec:ProfileSummary.
TN_parms	**Document** (optional) An IS document (IData object) that holds "hints" that Trading Networks uses when performing document recognition for a flat file document. See information about document gateway services in *webMethods Trading Networks Administrator's Guide* for details on providing recognition hints.

Usage Notes

- This service is protected by TNAdministrators ACL.
- To submit a document externally, use the wm.tn:receive service.
- If invoking this service from a Java program, in addition to returning *bizdoc*, *sender*, and *receiver* as IS documents (IData objects), the service returns *bizdoc* as an instance of com.wm.app.tn.doc.BizDocEnvelope and the returned *sender* and *receiver* as instances of com.wm.app.tn.profile.ProfileSummary.
- This service returns after Trading Networks completes processing for the document; that is, after Trading Networks executes the pre-processing and processing actions for the document. If the processing actions instruct Trading Networks to execute a service asynchronously, the asynchronously invoked service may not be complete.

wm.tn.doc.ff:validate

Validates the structure of a flat file document. This service uses the content of the *bizdoc*, settings from the TN document type and the pub.flatfile:convertToValues service to validate the flat file document.

Input Parameters

bizdoc	**Object** The flat file document to be validated. For the structure of *bizdoc*, see wm.tn.rec:BizDocEnvelope.

Output Parameters

errorCount	**String** The number of errors found while validating the document's structure.
errors	**String List** The errors found while validating the document's structure. If invoked from a Java service, *errors* is a String[ ].

Usage Notes

This service is invoked by Trading Networks if the matching TN document type and/or processing rule specifies that the document should be validated. You can invoke it explicitly if you have a *bizdoc*.

wm.tn.doc.xml:bizdocToRecord

Transforms a business document into an IS document (IData object), based on the IS document type blueprint associated with the TN document type (if any).

Input Parameters

bizdoc	**Document** The XML document from which to create an IS document (IData object). The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
processAsLargePart	**String** (Optional) Specifies how the service is to process the input XML document (bizdoc). Valid values are: ■ `true` - Uses large document handling to process the input document. Trading Networks writes large document content to hard disk drive space (called *tspace*) and keeps a pointer to the large document content in memory rather than the document content itself. Set to true when input documents are large or contain large parts.

- `false`- Default. Treat the document provided as input as normal size. That is, Trading Networks keeps the document's content in memory during processing.

Output Parameters

boundNode **Document** The content of the XML document bound into an IS document (IData object).

recordName **String** The fully-qualified name of the IS document type that was used to transform the XML document. This value is specified in the **Format as an IS document type** option on the Document Type Details page in My webMethods.

If the incoming Trading Networks document has only one element as a string or a document, then the output of the **wm.tn.doc.xml:bizdocToRecord** service depends on the value of the **Format as an IS document type** option in the Trading Networks Document Type.

- If the **Format as an IS document type** option is specified with the correct Document Type, then the **wm.tn.doc.xml:bizdocToRecord** service converts the incoming document correctly.
- If the **Format as an IS document type** option is empty, then the **wm.tn.doc.xml:bizdocToRecord** service converts the element to either a string or a document, and not a list.

If the incoming Trading Networks document has multiple elements as a string or a document and the **Format as an IS document type** option is empty, then the **wm.tn.doc.xml:bizdocToRecord** service converts the incoming document correctly.

Usage Notes

This service performs a function analogous to pub.xml:xmlNodeToDocument. For more information, see *webMethods Integration Server Built-In Services Reference*.

wm.tn.doc.xml:recordToBizdoc

Transforms an IS document (IData object) into an XML document and sends the resulting XML document into the document recognition engine to translate the XML document into a business document.

Input Parameters

boundNode	**Document** A IS document (IData object) containing the data to translate to a business document.
doctypeIdentifier	**String** (optional) Document type identifier for the resulting document. To determine the document type identifier of a document type, invoke the wm.tn.doctype:list service. The wm.tn.doctype:list service lists the names and IDs of all your TN document types.
rootTag	**String** (optional) Root tag for the resulting XML document.
htmlEncode	**String** (optional) Whether you want the leaf (String) data in the resulting XML document to be HTML-encoded (e.g., "<" replaced with "&lt"). Valid values are: ■ `true` - HTML-encode all leaf (String) data in the XML document. ■ `false` - Default. Do *not* HTML-encode leaf (String) data in the XML document.
recordName	**String** (optional) The fully-qualified name of the IS document type you want the service to use to guide the transformation of the XML document.
generateRequiredTags	**String** (optional) Indicates whether to create empty XML tags for all required elements of the specified IS document type (*recordName*) if there is no data available to complete them. Valid values are: ■ `true` - Create empty XML tags for required elements for which there is no data. ■ `false`- Default. Omit XML tags for required tags for which there is no data.
TN_parms	**Document** (optional) An IS document (IData object) that holds "hints" that Trading Networks uses when performing document recognition for a flat file document. For details on providing recognition hints, see the information about document gateway services in *webMethods Trading Networks Administrator's Guide*.

Output Parameters

bizdoc	**Document** The resulting document formatted as an IS document (IData object). For the structure of *bizdocs*, see wm.tn.rec:BizDocEnvelope.
TN_parms	**Document** (optional) An IS document (IData object) that holds "hints" that Trading Networks uses when performing document recognition for a flat file document. See information about document gateway services in

webMethods Trading Networks Administrator's Guide for details on providing recognition hints.

Usage Notes

- If you are invoking this service from a Java program, in addition to returning *bizdoc* as an IS document (IData object), the service returns *bizdoc* as an instance of com.wm.app.tn.doc.BizDocEnvelope.
- This service performs a function analogous to pub.xml:documentToXMLString. For more information about this service, see *webMethods Integration Server Built-In Services Reference*.
- This service is not applicable for large documents.

wm.tn.doc.xml:routeXml

Recognizes an XML document and submits it for processing.

This service does *not* check the identity of the sender against the currently logged in user. Only invoke this service from within processing rules or services; do *not* expose directly to trading partners. Trading partners should use wm.tn:receive.

Input Parameters

node	**Object** An XML document to process (must be an instance of com.wm.lang.xml.Document).
TN_parms	**Document** (optional) An IS document (IData object) that holds "hints" that Trading Networks uses when performing document recognition for a flat file document. For information about using recognition hints in gateway services, see *webMethods Trading Networks Administrator's Guide*.

Output Parameters

bizdoc	**Document** The XML document. For the structure of *bizdoc*, see wm.tn.rec:BizDocEnvelope.
sender	**Document** The profile summary for the sender of the document. For the structure of *sender*, see wm.tn.rec:ProfileSummary.
receiver	**Document** The profile summary for the receiver of the document. For the structure of *receiver*, see wm.tn.rec:ProfileSummary.
TN_parms	**Document** (optional) An IS document (IData object) that holds "hints" that Trading Networks uses when performing document recognition for a flat file document. For information about using recognition hints in gateway services, see *webMethods Trading Networks Administrator's Guide*.

Usage Notes

- This service is protected by the TNAdministrators ACL.
- To submit a document externally, use the wm.tn:receive service.
- If you are invoking this service from a Java program, in addition to returning *bizdoc*, *sender*, and *receiver* as IS documents (IData objects), the service returns *bizdoc* as an instance of com.wm.app.tn.doc.BizDocEnvelope and the returned *sender* and *receiver* as instances of com.wm.app.tn.profile.ProfileSummary.
- This service returns after Trading Networks completes processing for the document. That is, after Trading Networks executes the preprocessing and processing actions for the document. If the processing actions instruct Trading Networks to execute a service asynchronously, the asynchronously invoked service may not complete.

8 Docattr Folder

Overview

Use document attribute services (services in the wm.tn.docattr folder) to add, retrieve, enable, and disable document attributes. These services affect the definitions for document attributes that TN document types reference. The services do *not* affect the values of attributes in business documents.

Summary of Elements in this Folder

The elements that are available in this folder are listed in the following table:

Element	Description
wm.tn.docattr:add	Adds a new document attribute.
wm.tn.docattr:disable	Disables an existing document attribute.
wm.tn.docattr:enable	Enables an existing document attribute.
wm.tn.docattr:list	Retrieves the document attributes that are defined in the Trading Networks system.
wm.tn.docattr:listTypesForAttribute	Retrieves all TN document types that are associated with the specified document attribute.
wm.tn.docattr:setPersist	Sets whether or not you want a custom attribute saved to the database.
wm.tn.docattr:update	Updates an existing document attribute.
wm.tn.docattr:view	Retrieves a new document attribute.

wm.tn.docattr:add

Adds a new document attribute.

Input Parameters

attribute **Object** The document attribute to add. The document must be an instance of a subclass of com.wm.app.tn.doc.BizDocAttribute.

Output Parameters

updateCount **String** The number of attributes that the service added. Values indicate:

- `1` - The service added the attribute.
- `0` - The service did not add the attribute.

wm.tn.docattr:disable

Disables an existing document attribute.

Input Parameters

attributeId	**String** The internal unique identifier of the attribute to disable.

Output Parameters

updateCount	**String** The number of attributes that the service disabled. The following values indicate: - `1` - The service disabled the attribute. - `0` - The service did not disable the attribute.

Usage Notes

Trading Networks does *not* remove disabled document attributes from the Trading Networks database. Trading Networks does not use the attributes during document recognition. For example, disabled document attributes remain defined in TN document types that reference them. However, Trading Networks does not extract information for disabled attributes. If processing rules use the reference disabled custom attributes in the extended criteria, Trading Networks will be unable to obtain the value of the attribute from the document to match it against the value specified in the processing rule criteria.

wm.tn.docattr:enable

Enables an existing document attribute.

Input Parameters

attributeId	**String** The internal unique identifier of the attribute to enable.

Output Parameters

updateCount	**String** The number of attributes that the service enabled. Values indicate: - `1` - The service enabled the attribute. - `0` - The service did not enable the attribute.

wm.tn.docattr:list

Retrieves the document attributes that are defined in the Trading Networks system.

Input Parameters

refresh	**String** (optional) Whether Trading Networks refreshes its cache of document attributes before returning the list of document attributes. Valid values are: ■ `true` - Refresh the cache. ■ `false` - Default. Do not refresh the cache.
includeDeleted	**String** (optional) Whether to include disabled document attributes in the list of returned document attributes. Valid values are: ■ `true` - Return disabled document attributes in the list. ■ `false` - Default. Omit disabled document attributes from the list.

Output Parameters

attributeCount	**String** The number of attributes in the returned list.
attributes	**Document List** The list of returned document attributes. For the structure of each document attribute returned in *attributes*, see wm.tn.rec:BizDocAttribute.

Usage Notes

- Trading Networks caches document attribute definitions in memory on the server. A side effect of invoking this service with *refresh* set to `true` is that Trading Networks refreshes the cache. This is important if document attribute information has changed since Integration Server was started.
- If you are invoking this service from a Java program, in addition to returning the document attributes in *attributes* as IS documents (IData objects), the service returns the document attributes in *attributes* as instances of com.wm.app.tn.doc.BizDocAttribute.

wm.tn.docattr:listTypesForAttribute

Retrieves all TN document types that are associated with the specified document attribute.

Input Parameters

attributeId	**String** The internal unique identifier of the attribute for which to retrieve TN document types.

Output Parameters

typeCount	**String** The number of TN document types associated with the specified document attribute.
typeIdList	**String** A list of the internal identifiers for the TN document types associated with the specified document attribute.

wm.tn.docattr:setPersist

Sets whether to save a custom attribute to the database.

Input Parameters

attributeId	**String** The internal unique identifier of the attribute.
persist	**String** Whether Trading Networks saves the attribute. Valid values are: ■ `true` - Save the attribute. ■ `false` - Do not save the attribute.

Output Parameters

updateCount	**String** The status of the operation. Valid values are: ■ `1` - The service set the persist property on the attribute. ■ `0` - The service did not set the persist property on the attribute.

wm.tn.docattr:update

Updates an existing document attribute.

Input Parameters

attribute	**Object** The attribute that you want to update. You *must* specify an attribute that is already defined. The document must be an instance of a subclass of com.wm.app.tn.doc.BizDocAttribute.

Output Parameters

updateCount	**String** The number of attributes that the service updated. Values indicate: ■ `1` - The service updated the attribute. ■ `0` - The service did not update the attribute.

Usage Notes

You can only use this service to update existing attributes. To add a new attribute, use wm.tn.docattr:add. To get a listing of existing attributes that you can update, use wm.tn.docattr:list.

wm.tn.docattr:view

Retrieves a new document attribute.

Input Parameters

attribId	**String** (optional) The internal ID for the document attribute to retrieve.
attribName	**String** (optional) The name of the document attribute to retrieve.

Output Parameters

attrib	**Document** The requested document attribute, if it exists on the server (see wm.tn.rec:BizDocAttribute).

Usage Notes

- Either the *attribId* or *attribName* variable must be supplied.
- If you are invoking this service from a Java program, in addition to returning *attrib* as an IS document (IData object), the service returns *attrib* as an instance of com.wm.app.tn.doc.BizDocAttribute.

9 Doctype Folder

Overview

Use TN document type services (services in the wm.tn.doctype folder) to add, retrieve, enable, and disable TN document types.

Summary of Elements in this Folder

The elements that are available in this folder are listed in the following table:

Element	Description
wm.tn.doctype:add	Adds a new TN document type.
wm.tn.doctype:delete	Deletes an existing TN document type.
wm.tn.doctype:disable	Disables an existing TN document type.
wm.tn.doctype:enable	Enables an existing TN document type.
wm.tn.doctype:list	Retrieves the TN document types that are defined in Trading Networks.
wm.tn.doctype:update	Updates an existing TN document type.
wm.tn.doctype:view	Retrieves a single TN document type.

wm.tn.doctype:add

Adds a new TN document type.

Input Parameters

type

Object The TN document type that you want to add. The TN document type must be an instance of a subclass of com.wm.app.tn.doc.BizDocType.

If invoking from a Java program, the document must be an instance of a subclass of com.wm.app.tn.doc.BizDocType. Otherwise, *type* should have the structured defined by wm.tn.rec:BizDocType.

Output Parameters

updateCount

String The number of TN document types that the service added. The following values indicate:

- `1` - The service added the TN document type.
- `0` - The service did not add the TN document type.

Usage Notes

If you are using an OEM version of the Trading Networks, you cannot add new TN document types. This service will fail in an OEM environment.

wm.tn.doctype:delete

Deletes an existing TN document type. Optionally, it allows you delete the documents or activity logs associated with the TN document type you are deleting.

Important:
This operation is *not* recoverable.

Input Parameters

typeId **String** The internal unique identifier of the TN document type that you want to delete.

deleteDocuments **String** (optional) Whether you want this service to automatically delete the documents that are associated with this TN document type. The following values apply:

- `true` - Delete the documents associated with this TN document type; then it deletes the TN document type.
- `false` - Default. Do not delete the documents associated with this TN document type. If there are documents associated with this TN document type, this service throws an exception and aborts deleting the TN document type.

deleteLogs **String** (optional) Whether you want this service to automatically delete the activity logs that are associated with this TN document type. The following values apply:

- `true` - Delete the activity logs associated with this TN document type; If there are activity logs associated with this TN document type, this service deletes the activity logs.
- `false` - Default. Do *not* delete the activity logs associated with this TN document type. If there are activity logs associated with this TN document type, this service throws an exception and aborts deleting the document type.

Output Parameters

deleteCount **String** The number of TN document types that the service deleted. The following values indicate:

- `1` - The service deleted the TN document type.
- `0` - The service did not delete the TN document type.

Usage Notes

- You can use this service to delete any existing TN document types.
- This operation is not recoverable.
- If you set *deleteDocuments* to `true`, this service deletes all the documents associated with the TN document type.
- If you set *deleteDocuments* to `false` and there are documents associated with the TN document type, this service will throw an exception and abort deleting the TN document type.
- If you set *deleteDocuments* to `false` and there are no documents associated with the TN document type, this service deletes the TN document type.
- Before invoking this service, you can manually delete the documents associated with this TN document type by running the wm.tn.doc:deleteDocuments.

wm.tn.doctype:disable

Disables an existing TN document type.

Input Parameters

typeId	**String** The internal unique identifier of the TN document type that you want to disable.

Output Parameters

updateCount	**String** The number of TN document types that the service disabled. The following values indicate: - `1` - The service disabled the TN document type. - `0` - The service did not disable the TN document type.

Usage Notes

Trading Networks does *not* remove the disabled TN document types from the Trading Networks database. However, Trading Networks does not use the TN document types to recognize the type of documents.

wm.tn.doctype:enable

Enables an existing TN document type.

Input Parameters

typeId	**String** The internal unique identifier of the TN document type that you want to enable.

Output Parameters

updateCount	**String** The number of TN document types that the service enabled. The following values indicate: ■ `1` - The service enabled the TN document type. ■ `0` - The service did not enable the TN document type.

wm.tn.doctype:list

Retrieves the TN document types that are defined in the Trading Networks system.

Input Parameters

refresh	**String** (optional) Whether to refresh its cache of TN document types before returning the list of TN document types. Valid values are: ■ `true` - Refresh the cache. ■ `false` - Default. Do not refresh the cache.
includeDeleted	**String** (optional) Whether to include disabled TN document types in the returned list of TN document types. Valid values: ■ `true` - Return disabled TN document types in the list. ■ `false` - Default. Do not return disabled TN document types in the list.
includeHidden	**String** (optional) Whether you want the service to include hidden TN document types in the returned list of TN document types. Trading Networks system document types are hidden by default. Valid values are: ■ `true` - Return hidden TN document types in the list. ■ `false` - Default. Omit hidden TN document types from the list.

Output Parameters

typeCount	**String** The number of TN document types in the returned list.
types	**Document List** The list of summary information about the returned TN document types. For the structure of each TN document type returned in *types*, see wm.tn.rec:BizDocTypeSummary.

Usage Notes

Trading Networks caches TN document types in memory on the server. A side effect of invoking this service with *refresh* set to `true` is that Trading Networks refreshes the cache. This is important if TN document type information has changed since Integration Server started.

wm.tn.doctype:update

Updates an existing TN document type.

Input Parameters

type	**Object** The TN document type that you want to update. You *must* specify a TN document type that is already defined. The document must be an instance of a subclass of com.wm.app.tn.doc.BizDocType. Otherwise, *type* should have the structure defined by wm.tn.rec:BizDocType.

Output Parameters

updateCount	**String** The number of TN document types that the service updated. The following values indicate: ■ `1` - The service updated the TN document type. ■ `0` - The service did not update the TN document type.

Usage Notes

You can only use this service to update existing TN document types. To add a new TN document type, use wm.tn.doctype:add. To get a listing of existing types that you can update, use wm.tn.doctype:list.

wm.tn.doctype:view

Retrieves a single TN document type.

Input Parameters

typeId	**String** (optional) The internal unique identifier of the TN document type that you want to retrieve.
typeName	**String** (optional) The name of the TN document type that you want to retrieve.

Output Parameters

type	**Document** The requested TN document type, if it exists on Integration Server. For the structure of *type*, see wm.tn.rec:BizDocType.

Usage Notes

- You *must* supply either the *typeId* or *typeName* variable. If you supply both, Trading Networks uses *typeId*.
- If you are invoking this service from a Java program, in addition to returning *type* as an IS document (IData object), the service returns *type* as an instance of com.wm.app.tn.doc.BizDocType.

10 Enumerate Folder

Overview

Use enumeration services (services in the wm.tn.enumerate folder) to write clients that loop over large sets of results on the server. When the querying services (for example, wm.tn.query:documentQuery or wm.tn.query:eventQuery) are executed in paged mode, the services span a thread on the server side and return an ID to the client. The client can use this ID with services in the wm.tn.enumerate folder to navigate the result set from the query or to cancel the query.

Summary of Elements in this Folder

The elements that are available in this folder are listed in the following table:

Element	Description
wm.tn.enumerate:cancel	Cancels a running query on the Integration Server.
wm.tn.enumerate:deleteQueryResults	Deletes all Trading Networks query results from the webMethods repository.
wm.tn.enumerate:nth	Returns the nth page of an enumeration's data.
wm.tn.enumerate:unregister	Unregisters an enumeration; that is, it clears any server side query result object.

wm.tn.enumerate:cancel

Cancels a running query on the Integration Server.

Input Parameters

id	**String** ID of the query to cancel. Obtain *id* from the output of a service in the wm.tn.query folder.

Output Parameters

None.

wm.tn.enumerate:deleteQueryResults

Deletes all Trading Networks query results from the webMethods repository.

Input Parameters

None.

Output Parameters

None.

Usage Notes

When executing queries that return more rows than the tn.query.threshold parameter, Trading Networks saves the query result in the repository. These repository contexts start with the name TNQueryResults.

wm.tn.enumerate:nth

Returns the nth page of an enumeration's data. This service also provides some extra information about the server-side query thread.

Input Parameters

id	**String** The ID of the enumeration. Obtain *id* from the output of a service in the wm.tn.query folder.
pageNum	**String** The number of the page to return. (To return the first page, specify `1`.)

Output Parameters

resultCount	**String** The count of items in the result page.
results	**Object** The resulting page of data. It is a java.util.Vector.
rowsRead	**String** The number of rows that the query thread has read so far.
pagesRead	**String** The number of pages that the query thread has read so far.
threadRunning	**String** The state of the server-side query thread. The following values indicate: ■ `true` - The query thread is still running. ■ `false` - The query thread is completed.
totalPages	**String** The total number of pages in the result set. While the thread is still running, the service cannot determine this value. Therefore the service returns -1 while the query is running (*threadRunning* is `true`). When the query completes, (*threadRunning* is `false`), the service provides the total number of pages.
totalRows	**String** The total number of rows in the result set. While the thread is running, the service cannot determine this value. Therefore, the service

returns -1 while the query is running (*threadRunning* is `true`). When the query completes, (*threadRunning* is `false`), the service provides the total number of rows.

errors **Object** A java.util.Vector containing information about errors encountered reading the result set that is represented by this enumeration.

wm.tn.enumerate:unregister

Unregisters an enumeration; that is, it clears any server side query result object.

Input Parameters

id **String** The ID of the enumeration. Obtain *id* from the output of a service in the wm.tn.query folder.

Output Parameters

None.

Usage Notes

- Trading Networks registers an enumeration when you execute a service in the wm.tn.query folder.
- You should use this service to unregister a query after you are done using the results of the query.

11 Mime Folder

Overview

Use the MIME services (services in the wm.tn.mime folder) to work with MIME objects.

Unlike the services in the pub.mime folder (in the WmPublic package), the wm.tn.mime services support true streaming of data. As a result, these services never load the entire content of a MIME document into memory. This allows you to use these services to process or create MIME documents of any size, regardless of how much memory is available to the process in which the Integration Server is running.

The services in the wm.tn.mime folder are *not* compatible with the corresponding services provided in pub.mime folder. MIME data objects (for example, *mimeData*) that you create using the wm.tn.mime services will *not* function properly with pub.mime services, nor will MIME objects that you create using pub.mime services function with the wm.tn.mime services.

However, the pub.mime services can process the actual MIME documents that you create using the services in the wm.tn.mime folder, assuming there is sufficient memory to hold the entire MIME document. Similarly, the reverse is also true; that is, the wm.tn.mime services can process MIME documents you created using pub.mime.

Using the MIME Services to Send MIME Messages You Create

This section describes how to use the services in the wm.tn.mime folder to send a MIME message that you have created.

When you send a MIME message, the MIME message is one that you have created. When working with a MIME message that you create, use the wm.tn.mime:setDigestAlgorithm service to specify that you want to compute a message digest and to set the parameters for computing the digest. Use the wm.tn.mime:getDigest to retrieve the computed message digests.

The following are basic steps to follow to send a MIME message:

1. After creating the MIME objects that represent the message you want to send, invoke the "wm.tn.mime:setDigestAlgorithm" on page 150 for each body part for which you want to create a message digest.

2. To encrypt and/or sign a body part, use one of the following services: "wm.tn.mime:createSignedAndEncryptedData" on page 134, "wm.tn.mime:createEncryptedData" on page 131, or "wm.tn.mime:createSignedData" on page 136.

3. Invoke "wm.tn.mime:writeToStream" on page 155 to write the top-level MIME object to stream, all contained MIME objects, and to compute the message digests that you requested in "Using the MIME Services to Send MIME Messages You Create" on page 124. If you signed a body part, "wm.tn.mime:writeToStream" on page 155 only creates a message digest for the signed data of the body part.

4. To retrieve the message digests for the body parts that you requested, invoke "wm.tn.mime:getDigest" on page 140 for each body part

Using the MIME Services to Receive MIME Objects

This section describes how to use the services in the wm.tn.mime folder to receive a MIME message and process the received MIME message.

When you receive a MIME message that you need to parse, use the digest that wm.tn.mime:writeToStream computes. You specify the parameters for computing the digest in the input parameters to the wm.tn.mime:writeToStream service.

The following are basic steps to follow to receive a MIME message:

1. Invoke "wm.tn.mime:createMimeData" on page 132 to parse the InputStream into a MIME object.

2. Process the message, if necessary:

   - If the MIME object is encrypted, invoke wm.tn.mime:processEncryptedData to decrypt the data.
   - If the MIME object is signed, invoke wm.tn.mime:processSignedData to verify the digital signature.

     Note:
     To create digests of specified body parts rather than the entire MIME message, invoke wm.tn.mime:getBodyPartContent rather than processing the document.

3. Invoke "wm.tn.mime:writeToStream" on page 155 to write the MIME object. If you processed the document (decrypted or verified the digital signature), specify the processed message as the MIME object to write.

   If you want to compute the message digest of the document, be sure to set the input variable *createDigest* of the wm.tn.mime:writeToStream service to `yes`.

Summary of Elements in this Folder

The following table describes the elements that are available in this folder:

Element	Description
wm.tn.mime:addBodyPart	Adds a body part (header fields and content) to a specified MIME object.
wm.tn.mime:addMimeHeader	Adds one or more header fields to a specified MIME object.
wm.tn.mime:createCertsOnlyData	Generates a PKCS #7 certificate-only S/MIME entity from an array of specified certificates.
wm.tn.mime:createEncryptedData	Encrypts the contents of a MIME message.

Element	Description
wm.tn.mime:createMimeData	Parses a MIME message, creates a multipart mime message, or creates a single part mime message.
wm.tn.mime:createSignedAndEncryptedData	Digitally signs a MIME message, and then encrypts it.
wm.tn.mime:createSignedData	Digitally signs a MIME message.
wm.tn.mime:getBodyPartContent	Retrieves the content (payload) from the specified MIME object.
wm.tn.mime:getBodyPartHeader	Retrieves the headers from the specified body part of the specified MIME object.
wm.tn.mime:getContentType	Retrieves the value of the `Content-Type` message header from the specified MIME object.
wm.tn.mime:getDigest	Retrieves the message digest that the wm.tn.mime:writeToStream service computed.
wm.tn.mime:getMimeHeader	Retrieves the list of message headers from a specified MIME object.
wm.tn.mime:getNumParts	Retrieves the number of body parts in the specified MIME object.
wm.tn.mime:getParameterList	Retrieves the `Content-Type` parameters for the given MIME object.
wm.tn.mime:getPrimaryContentType	Retrieves the top-level portion (primary type) of a MIME object's `Content-Type` header value.
wm.tn.mime:getSharedInputStream	Retrieves an InputStream that implements the javax.mail.internet.SharedInputStream interface.
wm.tn.mime:getSize	Retrieves the size of this MIME object in bytes.
wm.tn.mime:getSubContentType	Retrieves the sub-type portion of a MIME object's `Content-Type` header value.
wm.tn.mime:processCertsOnlyData	Extracts the certificates from a PKCS #7 certificate-only S/MIME entity.
wm.tn.mime:processEncryptedData	Decrypts the specified encrypted MIME object and returns the decrypted MIME message.
wm.tn.mime:processSignedData	Processes a signed MIME object.
wm.tn.mime:removeHeader	Removes a specific mime header from the specified MIME object.

Element	Description
wm.tn.mime:resetMimeHeader	Resets all headers on this MIME object and optionally adds new headers.
wm.tn.mime:setDigestAlgorithm	Sets the digest algorithm that you want the wm.tn.mime:writeToStream service to use to compute a message digest for the specified MIME object when it writes the MIME object to a stream.
wm.tn.mime:sign	Creates a PKCS7 SignedData object.
wm.tn.mime:verify	Processes a digital signature to make sure that the specified data has not been changed.
wm.tn.mime:writeToStream	Writes the specified MIME object to a stream, and optionally allows you to create a message digest.

wm.tn.mime:addBodyPart

Adds a body part (header fields and content) to a specified MIME object.

Input Parameters

mimeData **Object** The MIME object to which to add a body part. You must create *mimeData* using the wm.tn.mime:createMimeData service.

content **Object or InputStream** The content that to add to the MIME object. Specify either an InputStream or another MIME object for *content*:

- Use an InputStream to add an ordinary payload.

  To support the creation of arbitrarily large mime messages, the InputStream should implement the interface javax.mail.internet.SharedInputStream. Use the wm.tn.mime:writeToStream service to obtain a SharedInputStream.

- Use a MIME object to add a payload that is itself a MIME message.

mimeHeader **Document** (optional) The header fields to add to the MIME object. Key names represent the names of the header fields. The values of the keys represent the values of the header fields. For example, if to add the following header fields:

```
X-Doctype: RFQ
X-Severity: 10
```

You would set the value of the keys of the *mimeHeader* as follows:

- Value of key `X-Doctype` as `RFQ`
- Value of key `X-Severity` as `10`

The wm.tn.mime:writeToStream service automatically inserts the following MIME headers when it generates the MIME message:

```
Message-ID
MIME-Version
```

If you set these values in *mimeHeader*, wm.tn.mime:writeToStream overwrites the values at run-time.

contenttype **String** (optional) The value of the `Content-type` header for this body part. You can also specify this value in *mimeHeader*. If you specify the value in both, this service uses the value in *mimeHeader*.

encoding **String** (optional) The value of the `Content-Transfer-Encoding` header for this body part. You can also specify this value in *mimeHeader*. If you specify the value in both, this service uses the value in *mimeHeader*.

Encoding determines how the service encodes the payload for transport. When you add a payload to *mimeData*, it should be in its original format. The wm.tn.mime:writeToStream service performs the encoding when it generates the final MIME message.

Specify one of the following values for encoding:

- `7bit` - Default. Content contains 7-bit, line-oriented text that needs no encoding.
- `8bit` - Content contains 8-bit, line-oriented text that needs no encoding.

  8bit is not recommended for messages to be transported through SMTP over the Internet because intervening mail servers that alter the data cannot accommodate 8-bit text. To safely transport 8-bit text, use quoted-printable encoding.
- `binary` - Content contains binary information that needs no encoding.

  Binary is not recommended for messages that are transported through SMTP over the Internet, because intervening mail servers that alter the data cannot accommodate binary data. To safely transport binary data, use base64 encoding.
- `quoted-printable` - Content contains 7 or 8-bit, line-oriented text to encode using the quoted-printable encoding scheme.
- `base64` - Content contains an arbitrary sequence of octets to encode using the base64 encoding scheme.
- `uuencode` - Contains an arbitrary sequence of octets to encode using the uuencode encoding scheme.

multipart	**String** (optional) If *mimeData* already contains one or more body parts, whether to append the body part being added, replace the body part(s) in *mimeData* with the body part being added, or throw an exception if body part(s) exist in *mimeData*. To construct a multipart document, set this parameter to append the body part. Valid values are: ■ `yes` - Default. Append a new body part to *mimeData*. ■ `no` - Replace the existing payload with the new body part or throw an exception, as determined by the value for replace.
replace	**String** (optional) If *mimeData* already contains a payload, whether to replace the existing payload or throw an exception. This service only ignores *replace* when *multipart* is `yes`. Valid values are: ■ `yes` - Default. Replace the existing payload with the new body part. ■ `no` - Throw an exception.
ignoreMimeVersion	**String** (optional) Add or ignore the Mime version header field in all the parts of a multi-part message. Valid values are: ■ `yes` - Ignores the Mime version header field. ■ `no` - Adds the Mime version header field

Output Parameters

None.

Usage Notes

- This service does not add output variables to the pipeline. Instead, it updates the contents of the existing *mimeData*.
- This service is not compatible with those in the pub.mime folder.
- The MIME objects that the services in the pub.mime folder creates will not work with this service.

wm.tn.mime:addMimeHeader

Adds one or more header fields to a specified MIME object.

Input Parameters

mimeData	**Object** The MIME object to which to add the header fields.

mimeHeader

Document (optional) The header fields to add to the MIME object. Key names represent the names of the header fields. The values of the keys represent the values of the header fields.

For example, if you want to add the following header fields:

```
X-Doctype: RFQ
X-Severity: 10
```

You would set the value of the keys of the *mimeHeader* as follows:

- Value of key `X-Doctype` as `RFQ`
- Value of key `X-Severity` as `10`

Be aware that the wm.tn.mime:writeToStream service automatically inserts the following MIME headers when it generates the MIME message:

```
Message-ID
MIME-Version
```

If you set these values in *mimeHeader*, the wm.tn.mime:writeToStream service overwrites the settings at run time.

Output Parameters

mimeData

Object The MIME object with the added header.

Usage Notes

This service is not compatible with those in the pub.mime folder. The MIME objects that the pub.mime:createMimeData service creates will not work with this service.

wm.tn.mime:createCertsOnlyData

Generates a PKCS #7 certificate-only S/MIME entity from an array of specified certificates.

Use this service to develop mechanisms for transmitting certificates and certificate chains to other parties.

Input Parameters

certificates

Object [ ] A list (a one-dimensional array) of byte arrays containing the certificates to encapsulate within the S/MIME entity.

Output Parameters

mimeData	**Object** A MIME object that contains the certificates only message.

Usage Notes

This service is not compatible with the services in the pub.mime or pub.smime folders. Only use the output of this service with services in the wm.tn.mime folder.

wm.tn.mime:createEncryptedData

Encrypts the contents of a MIME message.

Input Parameters

mimeSrc	**Object** The MIME object to encrypt.
recipientCerts	**Object [ ]** The X.509 certificates to use to encrypt the data. The certificates should be the certificates of the recipients for whom you are encrypting this message. Each element in the Object[] should contain a certificate for a single recipient (in the form of a byte array). **Note:** When you have multiple recipients, this service creates a single message that is encrypted for all recipients. It does not create a separate message for each recipient.
encryptionAlg	**String** The encryption algorithm to use. Specify one of the following values: `TripleDES`, `DES`, `AES`, or `RC2`. The default is `TripleDES`.
keyLength	**String** The length of the encryption key for RC2 and AES encryption. Specify one of the following values: ■ RC2 - `40`, `64`, or `128`. Default is `128`. ■ AES - `128`, `192`, or `256`. Default is `128` ■ DES - `128`. ■ TripleDES- `128`. **Note:** If you provide a value other than the one specified above, then Trading Networks uses `128` as the keyLength value.

Output Parameters

mimeData	**Object** A MIME object containing the encrypted message.

Usage Notes

This service is not compatible with the services in the pub.mime or pub.smime folders. Only use the output of this service with services in the wm.tn.mime folder.

wm.tn.mime:createMimeData

Parses a MIME message, creates a multipart mime message, or creates a single part mime message.

The MIME data object that this service creates is *not* compatible with the mime services in the pub.mime or pub.smime folders.

Input Parameters

input	**InputStream** (optional) The java.io.InputStream object that contains the MIME message to parse. Specify *input* to parse a MIME message.

mimeHeader	**Document** (optional) The header fields to add to the MIME object. Key names represent the names of the header fields. The values of the keys represent the values of the header fields.
	
	For example, to add the following header fields:
	
	`X-Doctype: RFQ`
	`X-Severity: 10`
	
	Set the value of the keys of the *mimeHeader* as follows:
	
	■ Value of key `X-Doctype` as `RFQ`
	
	■ Value of key `X-Severity` as `10`
	
	The wm.tn.mime:writeToStream service automatically inserts the following MIME headers when it generates the MIME message:
	
	`Message-ID`
	`MIME-Version`
	`Content-Type`
	`Content-Transfer-Encoding`
	
	If you set these values in *mimeHeader*, the wm.tn.mime:writeToStream service overwrites them at runtime.
	
	If you specify *mimeHeader*, you must also specify *subtype* or the service throws an exception.

Note:
This service ignores this parameter when you pass *input* to the service.

subType **String** (optional) The subtype component to use for the message's `Content-type` header. When you specify `related`, the service sets the message's `Content-type` header to "multipart/related." Specify *subtype* when you want to create a multipart MIME message.

Output Parameters

mimeData **Object** A MIME object. If you passed *input* to this service, *mimeData* contains the parsed MIME message. If you did not pass *input* to this service, *mimeData* is empty.

Note:
You cannot use this object with the services in the pub.mime folder.

encrypted **String** (optional) Whether *input* was an encrypted message. This string is present only if you specified a non-null value for *input*. Valid values are:

- `true` - Original message in *input* was encrypted.
- `false` - Original message in *input* was not encrypted.

signed **String** (optional) Whether *input* was a signed message. This string is present only if you specified a non-null value for *input*. Valid values are:

- `true` - Original message in *input* was signed.
- `false` - Original message in *input* was not signed.

certsOnly **String** (optional) Whether *input* contained only digital certificates. This string is present only if you specified a non-null value for *input*. Valid values are:

- `true` - Original message in *input* contained only digital certificates.
- `false` - Original message in *input* contained a regular payload.

Usage Notes

- You can use this service to parse a MIME message, create a multipart mime message, or create a single part mime message.
  - **To parse an existing MIME message**, set the input variables as follows:

For this input variable...	Specify...
input	The InputStream object that you want to parse. To parse an arbitrarily large MIME message, this InputStream object must implement the javax.mail.internet.SharedInputStream interface. Use the wm.tn.mime:writeToStream service to obtain an instance of a SharedInputStream.
mimeHeader	null
subtype	null

- **To create a multipart MIME message**, set the input variables as follows:

For this input variable...	Specify...
input	null
mimeHeader	Any additional headers.
subtype	The subtype of the message. When you specify `related`, the service sets the MIME message's `Content-type` header to "multipart/related."

- **To create a single part mime message**, all input parameters should be null.

- The *mimeData* object that this service produces is not compatible with *mimeData* objects produced by the service pub.mime:createMimeData. MIME objects that this service creates will not function with the services in the pub.mime folder.

wm.tn.mime:createSignedAndEncryptedData

Digitally signs a MIME message, and then encrypts it.

Input Parameters

mimeSrc	**Object** The MIME object to digitally sign and encrypt.
privKey	**Byte [ ]** The private key of the party signing the message.
signerCert	**Byte [ ]** The digital certificate of the party signing the message.
certificates	**Object [ ]** (optional) The certificate chain of the party signing the message. The chain must be in hierarchical order starting with the signer's certificate in first element (element zero).

The following shows a sample of a complete certificate chain if the signing party's certificate was signed by two intermediate certifying authorities (CAs). In the following list, for example, `0` is the element and Signer's certificate is the content.

- `0` - Signer's certificate
- `1` - Intermediary CA certificate
- `2` - Intermediary CA certificate
- `3` - Root CA certificate

Typically you should specify *certificates*. You can omit it only if the party receiving the message is able to process this signature without an accompanying certificate chain.

explicit

String (optional) Whether you want the service to generate an implicit or explicit signature. Valid values are:

- `true` - Default. Generate an explicit signature.
- `false` - Generate an implicit signature.

recipientCerts

Object [ ] The X.509 certificates to use to encrypt the data. The certificates should be the certificates of the recipients for whom you are encrypting this message. Each element in the Object[] should contain a certificate for a single recipient (in the form of a byte array).

Note:
When you have multiple recipients, this service creates a single message that is encrypted for all recipients. It does not create a separate message for each recipient.

encryptionAlg

String The encryption algorithm to use. Specify one of the following values: `TripleDES`, `DES`, or `RC2`. The default is `TripleDES`.

keyLength

String The length of the encryption key for RC2 encryption. Specify one of the following values:

- RC2 - `40`, `64`, or `128`. Default is `128`.
- AES - `128`, `192`, or `256`. Default is `128`
- DES - `128`.
- TripleDES- `128`.

Note:
If you provide a value other than the one specified above, then Trading Networks uses `128` as the keyLength value.

Output Parameters

mimeData	**Object** A MIME object containing the signed and encrypted message.

Usage Notes

This service is not compatible with the services in the pub.mime or pub.smime folders. Only use the output of this service with services in the wm.tn.mime folder.

wm.tn.mime:createSignedData

Digitally signs a MIME message.

Input Parameters

mimeSrc	**Object** The MIME object to encrypt.
privKey	**Byte [ ]** The private key of the party signing the message.
signerCert	**Byte [ ]** The digital certificate of the party signing the message.
certificates	**Object [ ]** (optional) The certificate chain of the party signing the message. The chain must be in hierarchical order starting with the signer's certificate in first element (element zero). The following shows a sample of a complete certificate chain if the signing party's certificate was signed by two intermediate certifying authorities (CAs). In the below list, for example, `0` is the element and Signer's certificate is the content. ■ `0` - Signer's certificate ■ `1` - Intermediary CA certificate ■ `2` - Intermediary CA certificate ■ `3` - Root CA certificate Typically you should specify *certificates*. You can omit it only if the party receiving the message is able to process this signature without an accompanying certificate chain.
explicit	**String** (optional) Whether to generate an implicit or explicit signature. Valid values are: ■ `true` Default. Generate an explicit signature. ■ `false` Generate an implicit signature.

recipientCerts	**Object [ ]** The X.509 certificates to use to encrypt the data. The certificates should be the certificates of the recipients for whom you are encrypting this message. Each element in the Object[] should contain a certificate for a single recipient (in the form of a byte array). **Note:** When you have multiple recipients, this service creates a single message that is encrypted for all recipients. It does not create a separate message for each recipient.

Output Parameters

mimeData	**Object** A MIME object containing the signed message.

Usage Notes

This service is not compatible with the services in the pub.mime or pub.smime folders. Only use the output of this service with services in the wm.tn.mime folder.

wm.tn.mime:getBodyPartContent

Retrieves the content (payload) from the specified MIME object.

Use this service for both single-part and multi-part messages. To retrieve content from a multi-part message, specify the part for which you want to retrieve content using the *index* or *contentID* variables.

Input Parameters

mimeData	**Object** The MIME object for which you want to retrieve the content.
index	**Integer** (optional) The index number of the body part with the content you want to retrieve. The first body part is index number zero. ■ To retrieve the content from a single-part message, set *index* to `0`, and do *not* specify *contentID*. ■ To retrieve the content for a specific part in a multi-part message, use either *index* or *contentID*. If you specify both, the service uses *contentID*.
contentID	**String** (optional) The value of the `Content-ID` header field of the body part for which you want to retrieve content. ■ To retrieve the content from a single-part message, do *not* use *contentID*; use *index*.

- To retrieve the content for a specific part in a multi-part message, use either *contentID* or *index*. If you specify both, the service uses *contentID*.

Output Parameters

content	**InputStream** An InputStream containing the content of the retrieved body part. The service removes all `Content-Transfer-Encodings`. Reading this stream consumes the MIME object, and you cannot use the wm.tn.mime:writeToStream service to re-create this object.
bodyPart	**Object** (optional) A MIME object containing the retrieved body part. If the MIME message is a single-part message, this object is the same as passed in *mimeData*.
encrypted	**String** (optional) Indicates whether *bodyPart* is an encrypted message. Valid values are: - `true` - The MIME object in *bodyPart* is encrypted. - `false` - The MIME object in *bodyPart* is not encrypted.
signed	**String** (optional) Indicates whether *bodyPart* is a signed message. Valid values are: - `true` - The MIME object in *bodyPart* is signed. - `false` - The MIME object in *bodyPart* is not signed.
certsOnly	**String** (optional) Indicates whether *bodyPart* contains digital certificates. Valid values are: - `true` - The MIME object in *bodyPart* contains only digital certificates. - `false` - The MIME object in *bodyPart* contains a regular payload.

Usage Notes

This service is not compatible with those in the pub.mime folder. The MIME objects that the pub.mime:createMimeData service creates will not work with this service.

wm.tn.mime:getBodyPartHeader

Retrieves the headers from the specified body part of the specified MIME object.

Input Parameters

mimeData	**Object** The MIME object for which you want to retrieve the message headers.

index **Integer** (optional) The index number of the body part that has the headers you want to retrieve. The first body part is index number zero.

- To retrieve the headers from a single-part message, set *index* to `0`, and do *not* specify *contentID*.
- To retrieve the headers for a specific body part in a multi-part message, use either *index* or *contentID*. If you specify both, the service uses *contentID*.

contentID **String** (optional) The value of the `Content-ID` header field of the body part from which you want to retrieve headers.

- To retrieve the headers from a single-part message, do *not* use *contentID*; use *index*.
- To retrieve the headers for a specific part in a multi-part message, use either *contentID* or *index*. If you specify both, the service uses *contentID*.

Output Parameters

mimeHeader **Document** The retrieved header fields. Key names represent the names of the header fields. The values of the keys represent the values of the header fields.

For example, if the original message contained the following header fields:

```
Content-Type: text/xml
X-Doctype: RFQ
X-Severity: 0
```

This service returns the following IS document (IData object):

- For key `Content-Type`, the value returned is `text/html`.
- For key `X-Doctype`, the value returned is `RFQ`.
- For key `X-Severity`, the value returned is `0`.

Usage Notes

This service is not compatible with those in the pub.mime folder. The MIME objects that the pub.mime:createMimeData service creates will not work with this service.

wm.tn.mime:getContentType

Retrieves the value of the`Content-Type` message header from the specified MIME object.

Input Parameters

mimeData	**Object** MIME object for which you want to retrieve the `Content-Type` message header.

Output Parameters

contentType	**String** A String containing the value of the MIME object's `Content-Type` header field. Note that this service returns only the media type and subtype portion of this header field's value. It does not return any parameters the value might include. For example, if the message's Content-Type header was: `content-type: text/plain;charset=UTF8` *contentType* would contain `text/plain`.

Usage Notes

This service is not compatible with those in the pub.mime folder. The MIME objects that the pub.mime:createMimeData service creates will not work with this service.

wm.tn.mime:getDigest

Retrieves the message digest that the wm.tn.mime:writeToStream service computed.

Input Parameters

mimeData	**Object** The MIME object for which you want to retrieve a message digest that you had computed using the wm.tn.mime:writeToStream service.

Output Parameters

messageDigest	**String** Base64 encoded message digest for the specified MIME object.

Usage Notes

- Use this service when sending a message that you created. For more information, see "Using the MIME Services to Send MIME Messages You Create" on page 124.
- You must invoke the wm.tn.mime:setDigestAlgorithm and wm.tn.mime:writeToStream services before invoking wm.tn.mime:getDigest.

- This service is not compatible with those in the pub.mime folder. The MIME objects that the pub.mime:createMimeData service creates will not work with this service.

wm.tn.mime:getMimeHeader

Retrieves the list of message headers from a specified MIME object.

Input Parameters

mimeData **Object** The MIME object for which you want to retrieve the list of message headers.

Output Parameters

mimeHeader **Document** The retrieved message headers. Key names represent the names of the header fields. The values of the keys represent the values of the header fields.

For example, if the original message contained the following header fields:

```
Content-Type: text/xml
X-Doctype: RFQ
X-Severity: 0
```

This service returns the following IS document (IData object):

- For key `Content-Type`, the value returned is `text/html`.
- For key `X-Doctype`, the value returned is `RFQ`.
- For key `X-Severity`, the value returned is `0`.

Usage Notes

This service is not compatible with those in the pub.mime folder. The MIME objects that the pub.mime:createMimeData service creates will not work with this service.

wm.tn.mime:getNumParts

Retrieves the number of body parts in the specified MIME object.

Input Parameters

mimeData **Object** The MIME object for which you want to retrieve the number of body parts.

Output Parameters

numParts **String** The number of body parts in the MIME object.

Usage Notes

This service is not compatible with those in the pub.mime folder. The MIME objects that the pub.mime:createMimeData service creates will not work with this service.

wm.tn.mime:getParameterList

Retrieves the `Content-Type` parameters for the given MIME object.

Input Parameters

mimeData **Object** A MIME object for which you want to retrieve the `Content-Type` parameters.

Output Parameters

parameters **Document** The retrieved parameters. Key names represent the names of the parameters fields. The values of the keys represent the values of the parameters.

```
content-type: text/plain;charset=UTF8; status=test
```

This service returns the following IS document (IData object):

- For key `charset`, the value returned is `UTF8`.
- For key `status`, the value returned is `test`.

Usage Notes

This service is not compatible with those in the pub.mime folder. The MIME objects that the pub.mime:createMimeDataservice creates will not work with this service.

wm.tn.mime:getPrimaryContentType

Retrieves the top-level portion (primary type) of a MIME object's `Content-Type` header value.

Input Parameters

mimeData	**Object** The MIME object for which you want to retrieve the value of the top-level portion (primary type) of the `Content-Type` header.

Output Parameters

primContentType	**String** The message's top-level (primary) content type. For example, if the message's `Content-Type` header was: `content-type: multipart/mixed` *primContentType* would contain `multipart`.

Usage Notes

This service is not compatible with those in the pub.mime folder. The MIME objects that the pub.mime:createMimeData service creates will not work with this service.

wm.tn.mime:getSharedInputStream

Retrieves an InputStream that implements the javax.mail.internet.SharedInputStream interface.

This allows for the parsing of arbitrarily large MIME objects.

Input Parameters

id	**String** The file name of the file from which the *inputStream* to read.
type	**String** The type of data source from which the *inputStream* to read Specify `file`. (Currently `file` is the only supported data source.)

Output Parameters

inputStream	**InputStream** A SharedInputStream that reads from the data source specified by *id* and *type*.

Usage Notes

This service is not compatible with those in the pub.mime folder. The MIME objects that the pub.mime:createMimeData service creates will not work with this service.

wm.tn.mime:getSize

Retrieves the size of this MIME object in bytes.

Input Parameters

mimeData	**Object** The MIME object of which you want to determine the size.

Output Parameters

partSize	**String** Approximate size of the MIME object in bytes. This service returns -1 if it could not determine the size.

Usage Notes

- The size returned in *partSize* might not be an exact measure of the content size and might or might not account for any transfer encoding of the content. The size is appropriate for display in a user interface to give the user an idea of the size of this part.
- This service is not compatible with those in the pub.mime folder. The MIME objects that the pub.mime:createMimeData service creates will not work with this service.

wm.tn.mime:getSubContentType

Retrieves the sub-type portion of a MIME object's `Content-Type` header value.

Input Parameters

mimeData	**Object** The MIME object for which you want to retrieve the sub-type portion of the `Content-Type` header.

Output Parameters

subContentType	**String** The message's subtype content type. For example, if the message's `Content-Type` header was: `content-type: multipart/mixed` *subContentType* would contain `mixed`.

Usage Notes

This service is not compatible with those in the pub.mime folder. The MIME objects that the pub.mime:createMimeData service creates will not work with this service.

wm.tn.mime:processCertsOnlyData

Extracts the certificates from a PKCS #7 certificate-only S/MIME entity.

Input Parameters

mimeData	**Object** The MIME message that contains certificate-only information.

Output Parameters

certificates	**Object [ ]** A list in which each element contains one of the extracted certificates.

Usage Notes

This service is not compatible with the services in the pub.mime or pub.smime folders. Only use the output of this service with services in the wm.tn.mime folder.

wm.tn.mime:processEncryptedData

Decrypts the specified encrypted MIME object and returns the decrypted MIME message.

Input Parameters

mimeSrc	**Object** A MIME object containing the encrypted data that you want decrypted.
recipientCert	**Byte [ ]** (optional) The digital certificate of the party receiving the message.
privKey	**Byte [ ]** (optional) The private key of the party receiving the message (i.e., the party whose public key was used to encrypt the message).
createDigest	**String** (optional) Whether to compute the message digest for the encrypted MIME message. Valid values are: - `yes` - Compute a message digest. - `no` - Default. Do *not* compute a message digest.

digestAlgorithm	**String** (optional) The algorithm to use to compute the digest if you specified `yes` for *createDigest*. You can specify one of the following values for *digestAlgorithm*: `SHA-1` or `MD5.SHA-1` is the default.

Output Parameters

mimeData	**Object** A MIME object containing the parsed contents of the decrypted message.
messageDigest	**String** Digest of the decrypted message.
encrypted	**String** (optional) Whether *mimeData* is an encrypted message. Valid values are: ■ `true` - The MIME object *mimeData* is encrypted. ■ `false` - The MIME object *mimeData* is not encrypted.
signed	**String** (optional) Whether *mimeData* is a signed message. Valid values are: ■ `true` - The MIME object *mimeData* is signed. ■ `false` - The MIME object *mimeData* is not signed.
certsOnly	**String** (optional) Whether *mimeData* contains only digital certificates. Valid values are: ■ `true` - MIME object *mimeData* contains only digital certificates. ■ `false` - MIME object *mimeData* contains a regular payload.

Usage Notes

- This service is not compatible with the services in the pub.mime or pub.smime folders. Only use the output of this service with services in the wm.tn.mime folder.
- *All* headers in the decrypted message are digested. If you need to compute the digest for selected headers, use the wm.tn.mime:writeToStream service.

wm.tn.mime:processSignedData

Processes a signed MIME object.

The service returns the message that was signed and attempts to verify the signature. If the service cannot verify the signature, it returns an error message explaining why the verification failed.

Input Parameters

mimeSrc **Object** A MIME object containing the signed data that you want processed.

signerCertChain **Object [ ]** (optional) The certificate chain of the party signing the message. The chain must be in hierarchical order starting with the signer's certificate in first element (element zero).

The following shows a sample of a complete certificate chain if the signing party's certificate was signed by two intermediate certifying authorities (CAs). In the below list, for example, `0` is the element and Signer's certificate is the content.

- `0`- Signer's certificate
- `1` - Intermediary CA certificate
- `2` - Intermediary CA certificate
- `3` - Root CA certificate

> **Note:**
> If the signer included the certificate chain with the digital signature, you do not need to supply *signerCertChain*.

createDigest **String** (optional) Whether to compute the message digest for the encrypted MIME message. Valid values are:

- `yes` - To compute a message digest.
- `no` - Default. To *not* compute a message digest.

digestAlgorithm **String** (optional) The algorithm to use to compute the digest if you specified `yes` for *createDigest*. You can specify one of the following values for *digestAlgorithm*: `SHA-1` or `MD5.SHA-1` is the default.

Output Parameters

mimeData **Object** A MIME object containing the parsed contents of the extracted MIME entity.

encrypted **String** (optional) Whether *mimeData* is an encrypted message. Valid values are:

- `true` - The MIME object in *mimeData* is encrypted.
- `false` - The MIME object in *mimeData* is not encrypted.

signed **String** (optional) Whether *mimeData* is a signed message. Valid values are:

- `true` - The MIME object in *mimeData* is signed.

- `false` - The MIME object in *mimeData* is not signed.

certsOnly **String** (optional) Whether *mimeData* contains only digital certificates. Valid values are:

- `true` - MIME object in *mimeData* contains only digital certificates.
- `false` - MIME object in *mimeData* contains a regular payload.

verify **String** Whether this service was able to successfully verify the digital signature of the signed message in *mimeSrc* with the public key supplied in the signer's certificate (*signerCertChain*). Valid values are:

- `true` - The service successfully verified the digital signature.
- `false` - The service was unable to successfully verify the digital signature. The service returns errorCode (1-4) and errorMessage to describe the error.

errorCode **String** (optional) A number (error code) that corresponds to the type of error that occurred while processing the digital signature. *errorMessage* contains a description of the error; see *errorMessage* for possible errors.

If the service does not encounter an error, it does not return *errorCode*.

errorMessage **String** (optional) A textual error message indicating the error that occurred while processing the digital signature.

The possible values returned in *errorCode* and *errorMessage* are as follows. In the below list, the numbers represent the *errorCode* and the text represents the *errorMessage*.For example, the first item indicates that for *errorCode* `1`, the possible *errorMessage* value is "Invalid signer certificate file information."

- `1`- Invalid signer certificate file information.
- `2` - Certificate at index, *i,* is not in recognizable format.
- `3` - Invalid certificate input at index, *i*.
- `4` - Signature cannot be verified.
- `5` - Expired certificate chain.
- `6` - Error in certificate chain.
- `7` - Untrusted certificate.

messageDigest **String** (optional) Digest of the signed message.

signerCert **Object** The certificate used to sign the message.

Usage Notes

- If *verify* is "false", *errorCode* and *errorMessage* indicate the error that caused the failure. The *errorCode* values 5 through 7 do not represent signature-validation failures and do not cause the *verify* flag to be set to "false".
- This service is not compatible with the services in the pub.mime or pub.smime folders. Only use the output of this service with services in the wm.tn.mime folder.
- *All* headers in the decrypted message are digested. If you need to compute the digest for selected headers, use the wm.tn.mime:writeToStream service.

wm.tn.mime:removeHeader

Removes a specific mime header from the specified MIME object.

Input Parameters

mimeData	**Object** MIME object containing the header you want to remove.
toRemove	**String** Header value you want to remove.

Output Parameters

None.

Usage Notes

This service is not compatible with those in the pub.mime folder. The MIME objects that the pub.mime:createMimeData service creates will not work with this service.

wm.tn.mime:resetMimeHeader

Resets all headers on this MIME object and optionally adds new headers.

Input Parameters

mimeData	**Object** The MIME object containing the headers that you want to reset.

mimeHeader	**Document** (optional) The header fields that you want to add to the MIME object. Key names represent the names of the header fields. The values of the keys represent the values of the header fields.
	
	For example, if you want to add the following header fields:
	
	`X-Doctype: RFQ`
	`X-Severity: 10`

You would set the values of the keys of the *mimeHeader* as follows:

- Value of key `X-Doctype` as `RFQ`
- Value of key `X-Severity` as `10`

Be aware that the wm.tn.mime:writeToStream service automatically inserts the following MIME headers when it generates the MIME message:

```
Message-ID
MIME-Version
Content-Type
Content-Transfer-Encoding
```

If you set these values in *mimeHeader*, the wm.tn.mime:writeToStream service overwrites them at run-time.

Output Parameters

None.

Usage Notes

This service is not compatible with those in the pub.mime folder. The MIME objects that the pub.mime:createMimeData service creates will not work with this service.

wm.tn.mime:setDigestAlgorithm

Sets the digest algorithm that you want the wm.tn.mime:writeToStream service to use to compute a message digest for the specified MIME object when it writes the MIME object to a stream.

Input Parameters

mimeData	**Object** The MIME object for which you want to compute a digest.
digestAlgorithm	**String** (optional) The algorithm to use to compute the digest. You can specify one of the following values for *digestAlgorithm*: `SHA-1` or `MD5.SHA-1` is the default.
digestHeader	**String** (optional) Whether to include the MIME headers when computing the message digest. Valid values are: - `yes`- Default. Include the headers when computing the message digest. - `no` - Do *not* include the headers when computing the message digest.
digestAllHeaders	**String** (optional) Whether to include all headers when computing the message digest or only those specified by the *headersToDigest* input

parameter. This parameter is used when you specify `yes` for the *digestHeader* input parameter. Valid values are:

- `yes` - Include *all* headers when computing the message digest. When you specify `yes`, this service ignores the *headersToDigest* input parameter.

  This service includes all headers for this MIME message when computing the message digest. The digest is computed based on the headers in the message at the time this service was invoked. If additional headers are added after you invoke this service, those headers will *not* be included in the final message digest.

- `no` - Default. Include only those headers specified by the *headersToDigest* input parameter when computing the message digest.

headersToDigest — **String [ ]** (optional) The headers to include in the message digest if *digestHeader* is `yes`. The default is { "Content-Type", "Content-Transfer-Encoding", "Content-Disposition" }.

The value that you specify for *headersToDigest* is *not* case sensitive. However, the order you specify the headers must match the order they appear in the message.

If no headers are specified (*headersToDigest* is empty), no headers will be digested.

Output Parameters

None.

Usage Notes

- Invoking this service on a MIME object indicates that you want the contents of this object to be digested when written to a stream using the wm.tn.mime:writeToStream service. To retrieve the message digest invoke the wm.tn.mime:getDigest service after calling wm.tn.mime:writeToStream. Do not alter the MIME object after invoking the wm.tn.mime:setDigestAlgorithm service because it will cause wm.tn.mime:writeToStream service to compute the message digest incorrectly.
- Use this service when sending a message that you created. For more information, see "Using the MIME Services to Send MIME Messages You Create" on page 124.
- This service is not compatible with those in the pub.mime folder. The MIME objects that the pub.mime:createMimeData service creates will not work with this service.

wm.tn.mime:sign

Creates a PKCS7 SignedData object.

The resulting signature is a Java byte[] that is a DER-encoded representation of the SignedData object as specified in PKCS7.

This service also allows for the creation of an explicit message digest. The service computes this digest separately from the one contained in the signed message. The default algorithm for this digest is SHA-1.

Input Parameters

signerInfo

IData [ ] An array containing information about a single signer of the document.

For each *signerInfoRecord*, one of the following is required:

- *certChain* that contains a certificate chain and private key
- *keyAlias* that references a certificate chain and private key in the key store

This IData object contains the following keys:

- *certChain* **Object [ ]** A X509 certificate of the signer. The certificate of the signer must be the first certificate in this chain; the root Certifying Authority (CA) certificate must be the last.

  This should be a Certificate[] or an array of byte[].
- *key* **Object** The private key that corresponds to the public key in the certificate of the signer. (That is, the first certificate in *certChain*). The service uses this key to digitally sign the data. The private key can be any asymmetric encryption key that is supported by the webMethods Integration Server; for example, DSA or RSA.

  This should be an instance of Java.security.PrivateKey or byte[].
- *keyAlias* **String** (optional) The alias that refers to the certificate chain and private key in the key store. This is not currently used.
- *hashAlgorithm* **String** The algorithm to use when computing the digest of the specified data. Specify either `SHA` or `MD5`.

data

InputStream or Byte [ ] The message to sign, which must be provided as a Java byte [].

This stream is fully read during the construction of the signature. Calls to read additional information from the stream after the execution of this service will fail.

detachedSignature

String Whether you want the created PKCS#7 object to contain the data that is digitally signed. A detached signature does not include the data. Valid values are:

- `true` - Create a detached signature that does *not* include the digitally signed data.
- `false` - Create a signature that includes the digitally signed data.

signatureStream — **OutputStream** (optional) OutputStream to which you want the signature written. If you do not specify *signatureStream* and *data* is a byte[], this service returns the signature as a byte[] in *signatureBytes*. If you do not specify *signatureStream* and *data* is an InputStream, this service throws an exception.

dataStream — **OutputStream** (optional) OutputStream to which you want the service to write the message contained in *data*. If you do not specify a stream for *dataStream*, the service discards the contents of *data*. This service *only* writes the message contained in *data* to this stream if you request a detached signature.

createDigest — **String** (optional) Whether you want the service to compute the message digest. Valid values are:

- `yes` - Compute a message digest.
- `no` - Default. Do not compute a message digest.

digestAlgorithm — **String** (optional) The algorithm to use to compute the digest if you specified `yes` for *createDigest*. You can specify one of the following values for *digestAlgorithm*: `SHA-1` or `MD5.SHA-1` is the default.

Output Parameters

bytesWritten — **String** Size (in bytes) of the signature.

signatureBytes — **Byte [ ]** (optional) If you did not specify *signatureStream* and *data* contained a byte[], *signatureBytes* contains the digital signature for the specified data.

messageDigest — **String** (optional) If *createDigest* is `yes` and you specified a valid value for *digestAlgorithm*, *messageDigest* contains a Base64 encoded message digest of the specified data.

Usage Notes

- This service provides the capability for multiple entities to sign the specified data.
- Each *signerInfo* block contained in the resulting signature contains the two authenticated attributes content type and a timestamp.

wm.tn.mime:verify

Processes a digital signature to make sure that the specified data has not been changed.

The signature input is the DER encoding of the PKCS#7 SignedData object. This service also allows for the creation of an explicit message digest.

Input Parameters

signature **InputStream** The signature to use for verifying that the data is unchanged. This stream will be fully read during the execution of this service.

data **InputStream** (optional) The data to verify for a detached signature. This service only uses *data* if *detachedSignature* is `true`. If present, this stream will be fully read during the execution of this service.

detachedSignature **String** Whether the *signature* contains a detached signature. Valid values are:

- `true` - The signature is detached.
- `false` - Default. The signature is *not* detached.

createDigest **String** (optional) Whether to compute the message digest for the message that was signed. Valid values are:

- `yes` - Compute a message digest.
- `no` - Default. Do *not* compute a message digest.

digestAlgorithm **String** (optional) The algorithm to use to compute the digest if you specified `yes` for *createDigest*. You can specify one of the following values for *digestAlgorithm*: `SHA-1` or `MD5.SHA-1` is the default.

outputStream **OutputStream** (optional) OutputStream where you want the contents of the signed message to be written. Specify if you want the contents written to an OutputStream.

If *signature* contains a detached signature, the value this service writes to the stream matches the contents of *data*. If *signature* does not contain a detached signature, this service writes the bytes that it signed to the stream.

Output Parameters

messageDigest **String** (optional) If *createDigest* is `yes` and you specified a valid value for *digestAlgorithm*, *messageDigest* contains a Base64 encoded message digest of the specified data.

signerInfo **Document List** Each element in this array contains information about a single signer of the signed data object.

The keys in each document are as follows:

- *certChain* **Object [ ]** A X509 certificate chain of a signer in Certificate[] format.
- *timeStamp* **Object** The time when the signer digitally signed the data; *timeStamp* is an instance of Java.util.Date.
- *trusted* **String** Whether the Integration Server trusts the certificate chain of the signer. Valid values are:
  - `true` - The certificate chain is trusted.
  - `false` - The certificate chain is trusted.
  - `unknown` - The certificate chain could not be reconstructed.
- *status* **String** Whether the signatures are intact within the signed data object.

  If the signature is intact, *status* is `verified`. Otherwise, the service returns an error message in *status* to indicate the problem.

wm.tn.mime:writeToStream

Writes the specified MIME object to a stream, and optionally allows you to create a message digest.

Input Parameters

mimeData **Object** The MIME object that you want written to a stream.

outputStream **OutputStream** Stream to which you want the contents of the MIME object written.

createDigest **String** (optional) Whether the service computes the message digest for the MIME message in *mimeData*. Valid values are:

- `yes` - Compute a message digest.
- `no` - Default. Do not compute a message digest.

digestAlgorithm **String** (optional) The algorithm to use to compute the digest if *createDigest* is `yes`. You can specify one of the following values for *digestAlgorithm*: `SHA-1` or `MD5.SHA-1` is the default.

digestHeader **String** (optional) Whether to include the MIME headers when computing the message digest if *createDigest* is `yes`. Valid values are:

- `yes` - Default. Include the headers when computing the message digest.
- `no` - Omit the headers when computing the message digest.

digestAllHeaders **String** (optional) Whether to include all headers when computing the message digest or only those specified by the *headersToDigest* input parameter. This parameter is used when you specify `yes` for the *digestHeader* input parameter. Valid values are:

- `yes` - Include *all* headers when computing the message digest. When you specify `yes`, this service ignores the *headersToDigest* input parameter.

  This service includes all headers for this MIME message when computing the message digest. The digest is computed based on the headers in the message at the time this service was invoked. If additional headers are added after you invoke this service, those headers will *not* be included in the final message digest.

- `no` - Default. Include only those headers specified by the *headersToDigest* input parameter when computing the message digest.

headersToDigest **String [ ]** (optional) The headers to include in the message digest if *digestHeader* is `yes`. The default is { "Content-Type", "Content-Transfer-Encoding", "Content-Disposition" }.

The value that you specify for *headersToDigest* is *not* case sensitive. However, the order you specify the headers must match the order they appear in the message.

If no headers are specified (*headersToDigest* is empty), no headers are digested.

Output Parameters

bytesWritten **String** Number of bytes written to *outputStream*.

messageDigest **String** (optional) If *digestHeader* is `yes`, *messageDigest* contains the Base64 encoded digest for the message written to *outputStream*.

Usage Notes

- Use this service to obtain a message digest from a MIME message that you are parsing. If you are creating a message, use the services wm.tn.mime:setDigestAlgorithm and wm.tn.mime:getDigest to compute digests.
- This service is not compatible with those in the pub.mime folder. The MIME objects that the pub.mime:createMimeData service creates will not work with this service.

12 Polling Folder

Overview

Use the polling services:

- When a partner wants to receive documents by polling Trading Networks without requiring Trading Networks to deliver the documents to the partner, directly.
- To allow a partner to find the list of documents available in the local environment or in the remote Trading Networks environment.
- To update the processing status of the documents received after polling.

Summary of Elements in this Folder

The following table describes the elements that are available in this folder:

Element	Description
wm.tn.polling:acceptDocument	Changes the processing status of the document received after polling.
wm.tn.polling:localCheck	Retrieves the list of pollable documents available for the partner in the local Trading Networks environment. The service retrieves the list of documents that are assigned to the partner during the user session.
wm.tn.polling:remoteCheck	Searches for the partner's corresponding B2B server where the pollable documents are stored and opens a client session. The service then retrieves the list of pollable documents and then processes the documents in the local Trading Networks environment.

wm.tn.polling:acceptDocument

Changes the processing status of the document received after polling.

Input Parameters

internalID **String** The internal ID of the document that Trading Networks generates for updating the processing status.

error **String** The status of the document based on its polling status. Valid values are:

- `true` To change the process status to “ACCEPTED,” when the polling is successful.

- `false` To change the process status to "ACCEPTED W/ ERR," when the polling is unsuccessful.

Output Parameters

None.

See Also

See also wm.tn.profile:addConnections and wm.tn.polling:remoteCheck.

wm.tn.polling:localCheck

Retrieves the list of pollable documents available for the partner in the local Trading Networks environment. The service retrieves the list of documents that are assigned to the partner during the user session.

Input Parameters

None.

Output Parameters

error	**String** The message of the error that occurs while retrieving the document list, if any.
resultCount	**String** The number of pollable documents available for the partner.
results	**String List** A string array that contains the internal IDs of all the pollable documents available for the partner. The results are sorted based on the date when the documents were created.

See Also

See also wm.tn.polling:acceptDocument and wm.tn.polling:remoteCheck.

wm.tn.polling:remoteCheck

Searches for the partner's corresponding B2B server where the pollable documents are stored and opens a client session. The service then retrieves the list of pollable documents and then processes the documents in the local Trading Networks environment.

Input Parameters

pid	**String** The internal ID of the document that Trading Networks generates.

Output Parameters

resultCount	**String List** A string array that contains the internal IDs of the documents that have been accepted and routed to the local environment.

See Also

See also wm.tn.polling:acceptDocument and wm.tn.profile:addConnections.

13 Profile Folder

Overview

Use profile services (services in the wm.tn.profile folder) to

- Add information to, update information in, and delete information from existing profiles in your trading network.
- Create new profiles by creating an empty profile, then adding profile components.

Summary of Elements in this Folder

- **Profile Management Services.** Use profile management services (services in the wm.tn.profile folder) to create and maintain information about your organization and the partners on your trading network.

The following table describes the elements that are available in this folder:

Element	Description
wm.tn.profile:addAddresses	Adds one or more new addresses to a partner's profile.
wm.tn.profile:addConnections	Adds one or more delivery methods to a partner's profile.
wm.tn.profile:addContacts	Adds one or more new contacts to a partner's profile.
wm.tn.profile:addIDs	Adds one or more external IDs to a partner's profile.
wm.tn.profile:addProfile	Adds a new partner profile to the trading network.
wm.tn.profile:addProfileGroups	Adds a new partner profile to a partner group.
wm.tn.profile:addUsers	Adds the specified user mappings to a Trading Networks profile.
wm.tn.profile:changeStatus	Changes the status of a partner in the trading network.
wm.tn.profile:deleteAddress	Deletes an address from a partner's profile.
wm.tn.profile:deleteConnection	Deletes a delivery method from a partner's profile.
wm.tn.profile:deleteConnectionOfPartner	
wm.tn.profile:deleteContact	Deletes a contact from a partner's profile.
wm.tn.profile:deleteID	Deletes an external ID from a partner's profile.

Element	Description
wm.tn.profile:deleteProfile	Deletes a partner profile.
wm.tn.profile:deleteProfileGroup	Deletes a partner from a profile group.
wm.tn.profile:deleteUser	Deletes a mapping of an user account to a Trading Networks profile.
wm.tn.profile:deleteUsers	Deletes all user account mappings associated with a Trading Networks profile.
wm.tn.profile:getExtendedFields	Retrieves a set of extended fields for a partner
wm.tn.profile:getExternalID	Retrieves an external ID for a trading partner.
wm.tn.profile:getExternalIDs	Retrieves all external IDs of the specified type for a trading partner.
wm.tn.profile:getHostProfile	Retrieves the profile for the local partner; that is, the organization that represents the host (or hub) of Trading Networks system.
wm.tn.profile:getInternalID	Retrieves the internal identifier for a trading partner.
wm.tn.profile:getInternalIDsByGroup	Retrieves Trading Networks generated internal IDs for all partners in the specified group.
wm.tn.profile:getInternalIDsForUser	Retrieves the list of all partners that are mapped to a specified user account.
wm.tn.profile:getPartnerBinary	Retrieves a binary value for a partner. Supply either a binaryTypeCode or a binaryTypeDescription. If values for both parameters are supplied, the binaryTypeDescription is ignored.
wm.tn.profile:getProfile	Retrieves the profile of a partner in your trading network.
wm.tn.profile:getProfileGroups	Retrieves a list of profile groups that a partner is a member of.
wm.tn.profile:getProfileMappings	Retrieves the profile name and profile ID mapping in your trading network.
wm.tn.profile:getProfileSummaries	Retrieves summary information about the partners in your trading network.
wm.tn.profile:getProfileSummary	Retrieves the profile summary information for a specified partner in your trading network.

Element	Description
wm.tn.profile:getUserProfiles	Retrieves the profiles of all partners mapped to the specified user account.
wm.tn.profile:getUserProfilesSummaries	Retrieves the profile summaries of all partners that are mapped to the specified user account.
wm.tn.profile:getUsersForPartner	Retrieves the users associated with a trading partner.
wm.tn.profile:setExtendedFields	Adds one or more extended fields to the profile of a partner that you specify, or changes one or more existing extended fields in the profile of the partner you specify.
wm.tn.profile:setPartnerBinary	Adds, updates, or deletes a binary value for a trading partner. Supply either a binaryTypeCode or a binaryTypeDescription. If values for both parameters are supplied, the binaryTypeDescription is ignored.
wm.tn.profile:undeleteProfile	Undeletes a profile that was previously deleted from the trading network.
wm.tn.profile:updateAddresses	Updates one or more addresses in a partner's profile.
wm.tn.profile:updateConnections	Updates one or more delivery method methods in a partner's profile.
wm.tn.profile:updateContacts	Updates one or more contacts in a partner's profile.
wm.tn.profile:updateCorporation	Updates the corporate information for a partner in the trading network.
wm.tn.profile:updateIDs	Updates one or more external IDs in a partner's profile.
wm.tn.profile:updateProfileGroups	Updates one or more profile groups that a partner is a member of.

- **Profile Creation Services.** Use the profile creation services to create empty profiles and profile components. You can then populate these objects by mapping or hard coding values into them. Save the new profile components to the Trading Networks database by passing them to the services in thewm.tn.profile folder. To update existing profile components, do *not* use the services in this folder. To update an existing profile component, use services in the wm.tn.profile folder to retrieve an existing profile, make the changes, and save your changes to the Trading Networks database.

The following table describes the elements that are available in this folder:

Element	Description
wm.tn.profile.create:newAddress	Creates an empty Address component for a profile.
wm.tn.profile.create:newContact	Creates an empty Contact component for a profile.
wm.tn.profile.create:newDelivery	Creates an empty delivery method component for a profile.
wm.tn.profile.create:newExtendedFields	Creates an extended field with no value.
wm.tn.profile.create:newExternalID	Creates an empty external ID component for a profile.
wm.tn.profile.create:newPrivateQueue	Creates a private queue for a trading partner.
wm.tn.profile.create:newProfile	Creates a Profile with an empty Corporation, no Delivery Methods, no Contacts, no external IDs, and no Addresses.

wm.tn.profile:addAddresses

Adds one or more new addresses to a partner's profile.

The service validates the addresses before saving them. If errors are found, the service does not save the addresses and returns the errors.

Input Parameters

partnerID	**String** The internal identifier for the trading partner.
addresses	**Object List** The addresses to add to the partner's (specified by *partnerID*) profile. The variable *addresses* must be an array of com.wm.app.tn.profile.Address instances.

Output Parameters

ids	**String List** The unique identifier that Trading Networks created for each address that the service added to the partner's profile.
errors	**String List** (optional) A string array that contains the errors found in the addresses, if any.

Usage Notes

- The wm.tn.profile:addAddresses service can either add the address as the partner's corporate address or as the address for one of the partner's contacts. To associate the address with the partner's organization, leave the ContactID field of the com.wm.app.tn.profile.Address instance empty. To

associate the address with a contact, set the ContactID field of the Address object to the internal identifier of the contact.

- A partner's organization can have multiple addresses. A contact can have either no addresses or one address.

wm.tn.profile:addConnections

Adds one or more delivery methods to a partner's profile. The service validates the delivery method information before saving it. If errors are found, the service does not save the delivery method information and returns the errors.

Input Parameters

partnerID	**String** The internal identifier for the trading partner.
connections	**Object List** The delivery methods to add to the partner's (specified by *partnerID*) profile. The variable must be an array of com.wm.app.tn.profile.Destination instances.

Output Parameters

ids	**String List** The unique identifier that Trading Networks created for each delivery method that the service added to the partner's profile.
errors	**String List** (optional) A string array that contains the errors found in the delivery methods, if any.

Usage Notes

You can associate, at most, one of each of the following types of com.wm.app.tn.profile. Destination instances with a partner:

- Primary HTTP
- Secondary HTTP
- Primary HTTPS
- Secondary HTTPS
- Primary SMTP
- Secondary SMTP
- Primary FTP
- Secondary FTP
- A delivery method that you created using HTTP, FTP, E-mail, and so on

- A custom delivery method that you created and registered using wm.tn.delivery:registerService

wm.tn.profile:addContacts

Adds one or more new contacts to a partner's profile. The service validates the contacts before saving them. If errors are found, the service does not save the contacts and returns the errors.

Input Parameters

partnerID	**String** The internal identifier for the trading partner.
contacts	**Object List** The contacts to add to the partner's (specified by *partnerID*) profile. The variable *contacts* must be an array of com.wm.app.tn.profile.Contact instances.

Output Parameters

ids	**String List** The unique identifier that Trading Networks created for each contact that the service added to the partner's profile.
errors	**String List** (optional) A string array that contains the errors found in the contacts, if any.

wm.tn.profile:addIDs

Adds one or more external IDs to a partner's profile. An external ID is an ID type within a document that identifies a corporation, for example, a D-U-N-S® number. The service validates the external IDs before saving them. If errors are found, the service does not save the external IDs and returns the errors.

Input Parameters

partnerID	**String** The internal identifier for the trading partner.
ids	**Object List** The external IDs to add to the partner's (specified by *partnerID*) profile. The variable *ids* must be an array of com.wm.app.tn.profile.ID instances.

Output Parameters

idIDs	**String List** A java.util.Vector that holds the internal identifier for each external ID that was added. This is the primary key to the *partnerID* table in the Trading Networks database.

username	**String** (optional) If Trading Networks created an Integration Server user account, this is the user name for the user account.
password	**String** (optional) If Trading Networks created an Integration Server user account, this is the password of the user account.
errors	**String List** (optional) A string array that contains errors found in the external IDs, if any.

wm.tn.profile:addProfile

Adds a new partner profile to the trading network. The service validates the profile and saves it with an Inactive status.

Input Parameters

profile	**Object** The profile to add to the trading network. The variable *profile* must be an instance of com.wm.app.tn.profile.Profile.
security	**Document** (optional) Leave null. This field is for internal use only and using this field can result in database errors.

Output Parameters

partnerID	**String** If the operation was successful, the internal ID created for this partner.
username	**String** (optional) If Trading Networks created an Integration Server user account, this is the user name for the user account.
password	**String** (optional) If Trading Networks created an Integration Server user account, this is the password of the user account.
errors	**String List** (optional) A string array that contains the errors found in the profile, if any.

Usage Notes

This service saves the profile regardless of whether errors are returned in *errors*.

wm.tn.profile:addProfileGroups

Adds a new partner profile to a partner group.

Input Parameters

partnerId	**String** The internal identifier for the trading partner profile.
profileGroupIds	**String List** The partner group IDs to which to add this partner.

Output Parameters

None.

wm.tn.profile:addUsers

Adds the specified user mappings to a Trading Networks profile. When the service is executed through a client, the service creates the Integration Server user account if it does not already exist and associates the user account with the profile.

Input Parameters

partnerId	**String** The internal identifier for the trading partner profile.
users	**String List** The users to add to the profile. Specify each user to associate with the profile as a separate String in the String List. The user name can be up to 128 characters and cannot contain spaces.

Output Parameters

accounts	**Document List** The user accounts that were added to the profile. Each document contains the following keys: ■ *user* The user name for the user account. ■ *password* The password that corresponds to the user account.

wm.tn.profile:changeStatus

Changes the status of a partner in the trading network. Use this service to activate and deactivate trading partners. When activating a partner, the partner's profile is validated. If errors are found, the service does not activate the partner and returns the errors.

Input Parameters

partnerID	**String** The internal identifier for the trading partner for which to change the status.

status **String** The new status for the partner. Valid values are:

- `Active` Activates the partner.
- `Inactive` Deactivates the partner.

Output Parameters

errors **String List** (optional) A string array that contains the errors that occurred when activating a partner, if any.

wm.tn.profile:deleteAddress

Deletes an address from a partner's profile.

Input Parameters

partnerID **String** The internal identifier of the trading partner's associated with the address to delete.

addressID **String** The internal identifier of the address to delete.

Output Parameters

None.

wm.tn.profile:deleteConnection

Deletes a delivery method from a partner's profile.

Input Parameters

destinationID **String** Internal identifier of the delivery method to delete from the partner's profile.

Output Parameters

errors **String List** (optional) Errors that Trading Networks encountered while deleting the delivery method. There is one string in the string list for each error. If *errors* contains a non-null value, Trading Networks did *not* perform the delete action.

wm.tn.profile:deleteConnectionOfPartner

If the delivery methods to be deleted are associated with other partners, they will not be deleted.

Input Parameters

destinationID	**String** Internal identifier of the delivery method to delete from the partner's profile.
partnerID	**String** The internal identifier of the trading partner where the delivery method is associated.

Output Parameters

errors	**String List** (optional) Errors that Trading Networks encountered while deleting the connection to a partners profile. There is one string in the string list for each error. If *errors* contains a non-null value, Trading Networks did *not* perform the delete action.

wm.tn.profile:deleteContact

Deletes a contact from a partner's profile.

Input Parameters

partnerID	**String** The internal identifier of the trading partner associated with the contact to delete.
contactID	**String** The internal identifier of the contact to delete from the partner's profile.

Output Parameters

None.

wm.tn.profile:deleteID

Deletes an external ID from a partner's profile.

Input Parameters

id	**String** The internal identifier of the partner external ID to delete.

Output Parameters

errors	**String List** (optional) Errors that Trading Networks encountered while deleting the partner external ID. There is one string in the string list for each error. If *errors* contains a non-null value, Trading Networks did *not* perform the delete action.

wm.tn.profile:deleteProfile

Deletes a partner profile.

Input Parameters

partnerID	**String** The internal identifier for the trading partner to delete.

Output Parameters

None.

Usage Notes

- You cannot physically remove a partner from the database using built-in services. This can be done only by issuing SQL commands directly to the Trading Networks database.
- When the profile is deleted, if the profile was mapped to any user accounts that Trading Networks created and those user accounts are not mapped to any other profile, the service deletes those user accounts.

wm.tn.profile:deleteProfileGroup

Deletes a partner from a profile group.

Input Parameters

partnerId	**String** The internal identifier for the trading partner.
profileGroupId	**String** The internal identifier of the profile group ID to delete for the partner.

Output Parameters

None.

wm.tn.profile:deleteUser

Deletes a mapping of a user account to a Trading Networks profile.

If Trading Networks created the user account and the user account is not associated with any other profile, the service deletes the user account as well.

Input Parameters

username	**String** The user name of the user account mapping to delete from the profile.
partnerId	**String** The internal identifier for the trading partner profile.

Output Parameters

None.

wm.tn.profile:deleteUsers

Deletes all user account mappings associated with a Trading Networks profile. If Trading Networks created a user account for a mapping being deleted and the user account is not associated with any other profile, the service deletes the user accounts as well.

Input Parameters

partnerId	**String** The internal identifier for the trading partner profile.

Output Parameters

None.

wm.tn.profile:getExtendedFields

Retrieves a set of extended fields for a partner.

Input Parameters

partnerID	**String** The internal identifier of the trading partner for which to retrieve extended fields.
group	**String** The set of extended fields to retrieve. Specify the name of the field group associated with the extended files to retrieve. You can specify one of the following standard field groups or the name of a user-defined group that you defined using the wm.tn.dictionary:addFieldGroup service:

- `Corporation` - To retrieve Corporation extended fields.
- `Contact` - To retrieve Contact extended fields.
- `Delivery` - To retrieve Delivery extended fields.
- `ID` - To retrieve IDs extended fields.
- `Address` - To retrieve Addresses extended fields.
- `Custom` - To retrieve Custom extended fields.

Output Parameters

fields	**Document List** The extended fields that you requested to retrieve. Each extended field in the variable *fields* is in the format wm.tn.rec:Field. For the format, see wm.tn.rec:Field.

Usage Notes

There is, at most, one set of extended fields for each of the groups identified by *group*. Although a partner might have multiple contacts, there is only one set of extended fields for the Contact group. Similarly, there is one set of extended fields for the Delivery group, ID group, and Address group even though each of these groups can have multiple members.

wm.tn.profile:getExternalID

Retrieves an external ID for a trading partner.

Input Parameters

partnerID	**String** The internal identifier for a trading partner for which to retrieve an external ID.
idType	**String** Deprecated. Use *idTypeDesc* instead.
idTypeDesc	**String** The type of the external ID to retrieve. The value must match one of the values in the Description column of the IDType table in the Trading Networks database (for example, `DUNS`; `DUNS+4`; `EULER`, `User Defined 3`). Supply a value for either *idTypeDesc* or *idTypeCode.*
idTypeCode	**String** The type of the external ID to retrieve. The value must match one of the values in the Type column of the IDType table in the Trading Networks database (for example, 1, 2, 3). Supply a value for either *idTypeCode* or *idTypeDesc*.

Output Parameters

id	**String** The requested external ID.

Usage Notes

- When exchanging documents, partners typically identify themselves within a document using some well-known ID scheme, such as a D-U-N-S number. If you know a partner's internal ID, use this service to get the external ID you need.
- To retrieve the ID types from the IDType table in the Trading Networks database, use the wm.tn.dictionary:addIDType service. The wm.tn.dictionary:addIDType service returns a Hashtable of all types of IDs known to Trading Networks. The value of *idType* must be one of the keys from this Hashtable. For more information, see wm.tn.dictionary:addIDType.
- Supply a value for either *idTypeDesc* or *idTypeCode*. If you supply values for both of these parameters, the service uses *idTypeCode* and ignores *idTypeDesc*.
- As of version 4.6, Trading Networks allows you to store multiple IDs of the same type for a partner. If the partner identified by the *partnerID* parameter has more than one ID of the requested type, this service will return the first ID in the database, which is usually the first one you entered. To retrieve all IDs of a particular type for a partner, use the wm.tn.profile:getExternalIDs service.

wm.tn.profile:getExternalIDs

Retrieves all external IDs of the specified type for a trading partner.

Input Parameters

partnerID	**String** The internal identifier for a trading partner for which to retrieve external IDs.
idTypeDesc	**String** The type of external IDs to retrieve. The value must match one of the values in the Description column of the IDType table in the Trading Networks database (for example, `DUNS`; `DUNS+4`; `EULER`, `User Defined 3`). Supply a value for either *idTypeCode* or *idTypeDesc.*
idTypeCode	**String** The type of external IDs to retrieve. The value must match one of the values in the Type column of the IDType table in the Trading Networks database (for example, 1, 2, 3). Supply a value for either *idTypeCode* or *idTypeDesc*.

Output Parameters

ids	**String List** The requested external IDs.

Usage Notes

- When exchanging documents, partners typically identify themselves within a document using some well-known ID scheme, such as a D-U-N-S number. If you know a partner's internal ID, use this service to get the external ID you need.
- To retrieve the ID types from the IDType table in the Trading Networks database, use the wm.tn.dictionary:getIDTypes service. The wm.tn.dictionary:getIDTypes service returns a Hashtable of all types of IDs known to Trading Networks. The value of *idType* must be one of the keys from this Hashtable. For more information, see wm.tn.dictionary:getIDTypes.
- Supply a value for either *idTypeCode* or *idTypeDesc*. If you supply values for both of these parameters, the service uses *idTypeCode* and ignores *idTypeDesc*.

wm.tn.profile:getHostProfile

Retrieves the profile for the local partner; that is, the organization that represents the host (or hub) of Trading Networks system.

Input Parameters

None.

Output Parameters

profile	**Document** The profile for the local partner. For the structure of the *profile*, see wm.tn.rec:Profile.

Usage Notes

The local partner (My Enterprise) must be created before you can use this service to retrieve it.

wm.tn.profile:getInternalID

Retrieves the internal identifier for a trading partner.

Input Parameters

id	**String** The external ID for a partner.
idType	**String** Deprecated. Use *idTypeDesc* instead.
idTypeDesc	**String** The type of the external ID in the *id* parameter. The value must match one of the values in the Description column of the IDType table in the Trading Networks database (for example, `DUNS`; `DUNS+4`; `EULER`, `User Defined`

	`3`, `User Defined 2`, `User Defined 1`, `webMethods Internal`, `Mutually defined`). Supply a value for either *idTypeDesc* or *idTypeCode*.
idTypeCode	**String** The type of the external ID in the *id* parameter. The value must match one of the values in the Type column of the IDType table in the Trading Networks database (for example, 1, 2, 3). Supply a value for either *idTypeCode* or *idTypeDesc*.

Output Parameters

partnerID	**String** The internal identifier for a trading.

Usage Notes

- When exchanging documents, partners typically identify themselves within a document using some well-known ID scheme, such as a D-U-N-S number. If you have a partner's external ID from a business document, use this service to get the internal identifier; that is, the identifier that Trading Networks generated for the partner.
- To retrieve the ID types from the IDType table in the Trading Networks database, use the wm.tn.dictionary:getIDTypes service. The wm.tn.dictionary:getIDTypes service returns a Hashtable of all types of IDs known to Trading Networks. The value of *idType* must be one of the keys from this Hashtable. For more information, see wm.tn.dictionary:getIDTypes.
- Supply a value for either *idTypeDesc* or *idTypeCode*. If you supply values for both of these parameters, the service uses *idTypeCode* and ignores *idTypeDesc*.

wm.tn.profile:getInternalIDsByGroup

Retrieves Trading Networks generated internal IDs for all partners in the specified group.

Input Parameters

groupId	**String** Group ID of the partners in the specified group.

Output Parameters

partnerIDs	**Array** Array of internal partner IDs.

wm.tn.profile:getInternalIDsForUser

Retrieves the list of all partners that are mapped to a specified user account.

Input Parameters

username	**String** The user name of a user account. The service returns the list of partners that are mapped to the specified user account.

Output Parameters

partners	**String List** Internal IDs of the trading partners that are mapped to the user account specified by *username*.

wm.tn.profile:getPartnerBinary

Retrieves a binary value for a partner. Supply either a binaryTypeCode or a binaryTypeDescription. If values for both parameters are supplied, the binaryTypeDescription is ignored.

Input Parameters

partnerID	**String** (optional) The internal identifier for the trading partner.
binaryTypeCode	**String** (optional) A string that contains the numeric code for the type of binary to be retrieved. This value must match the Type column of the PartnerBinary table. Supply a value for binaryTypeCode or for binaryTypeDescription, but not both.
binaryTypeDescription	**String** (optional) A string that describes the type of binary to be retrieved. This value must match the Description column of the PartnerBinary table. Supply a value for binaryTypeDescription or for binaryTypeCode, but not both.

Output Parameters

binary	**Object** A byte array that contains the requested binary value.

wm.tn.profile:getProfile

Retrieves the profile of a partner in your trading network.

Input Parameters

partnerID	**String** The internal identifier of the trading partner's profile to retrieve.

Output Parameters

profile	**Document** The partner's profile that was retrieved. For the structure of the *profile*, see wm.tn.rec:Profile.

Usage Notes

- If there is no partner on the network with the *partnerID* that you specified, the service returns no value.
- If you do not know the internal identifier for the partner (*partnerID*), use wm.tn.profile:getProfileSummaries that returns frequently used fields including the partner identifier for all partners in the trading network. For more information, see wm.tn.profile:getProfileSummaries.

wm.tn.profile:getProfileGroups

Retrieves a list of profile groups that a partner is a member of.

Input Parameters

partnerID	**String** The internal identifier for the trading partner.

Output Parameters

profileGroupIds	**String List** A list of profile group IDs of which this partner is a member.

wm.tn.profile:getProfileMappings

Retrieves the profile name and profile ID mapping in your trading network.

Input Parameters

deleted	**String** (optional) Whether to retrieve deleted profiles. Valid values are: - `true` To retrieve deleted profiles. - `false` Default. To *not* retrieve non-deleted profiles.
refresh	**String** (optional) Whether you want Trading Networks to refresh the profile cache on the server before retrieving profiles. Valid values are: - `true` - Refreshes the profile cache on the server before retrieving the profiles.

- `false` - Default. Does not refresh the profile cache on the server before retrieving the profiles.

Output Parameters

NameIdMap	**Object** Profile names and profile IDs as key/value pairs, where name is the key and ID is the value. It is a type of java.util.HashMap.
IdNameMap	**HashMap** Profile IDs and profile names as key/value pairs, where ID is the key and name is the value. It is a type of java.util.HashMap.
self	An enterprise profile. It is a type of wm.tn.rec:ProfileSummary.
unknown	An unknown profile. It is a type of wm.tn.rec:ProfileSummary.

wm.tn.profile:getProfileSummaries

Retrieves summary information about the partners in your trading network.

Input Parameters

deleted — **String** (optional) Whether to retrieve deleted profile summaries. Valid values are:

- `true` To retrieve deleted profile summaries.
- `false` Default. To *not* retrieve non-deleted profile summaries.

refresh — **String** (optional) Whether you want Trading Networks to refresh the profile summaries cache on the server before retrieving profile summaries. Valid values are:

- `true` Refreshes the cache of profile summaries on the server before retrieving the profile summaries.
- `false` Default. Retrieves the cached profile summaries on the server.

Output Parameters

profiles — **Document List** A list of wm.tn.rec:ProfileSummary IS document types (IData objects). For the structure of each IS document in the *profiles* IS document list, see wm.tn.rec:ProfileSummary.

Usage Notes

If the profiles on your network are maintained through services in the wm.tn.profile folder, you do not need to specify *refresh* equal to `true`. If you use SQL commands against the Trading Networks database to manipulate profiles, invoke wm.tn.profile:getProfileSummaries with *refresh* set to `true` to refresh the internal cache of profile summary information.

wm.tn.profile:getProfileSummary

Retrieves the profile summary information for a specified partner in your trading network.

Input Parameters

partnerID	**String** The internal identifier to retrieve of the trading partner's profile summary.
refresh	**String** (optional) Whether Trading Networks refreshes the cache of profile summaries on the server before retrieving profile summaries.

Output Parameters

profile	**Document** The profile summary. For the structure of *profile*, see wm.tn.rec:ProfileSummary.

Usage Notes

If there is no partner in the trading network with the *partnerID* that you specified, the service does not return a value.

wm.tn.profile:getUserProfiles

Retrieves the profiles of all partners mapped to the specified user account.

Input Parameters

username	**String** The user name of the user account for which to retrieve profiles.

Output Parameters

profiles	**Document List** The partner profiles that the service retrieved. For the structure of a profile, see wm.tn.rec:Profile.

Usage Notes

If there are no partners in the trading network that are associated with the specified user name (for example because the user account was deleted), the service returns no value.

wm.tn.profile:getUserProfilesSummaries

Retrieves the profile summaries of all partners that are mapped to the specified user account.

Input Parameters

username	**String** The user name of the user account for which to retrieve profile summaries.
refresh	**String** (optional) Whether Trading Networks refreshes the cache of profile summaries on the server before retrieving profile summaries.

Output Parameters

profiles	**Document List** The partner profiles that the service retrieved. For the structure a profile summary, see wm.tn.rec:ProfileSummary.

Usage Notes

If there are no partners in the trading network that are associated with the specified user name (for example because the user account was deleted), the service returns no value.

wm.tn.profile:getUsersForPartner

Retrieves the users associated with a trading partner.

Input Parameters

partnerID	**String** The internal identifier for the trading partner.

Output Parameters

users	**Array** List of user names associated with a partner.

wm.tn.profile:setExtendedFields

Adds one or more extended fields to the profile of a partner that you specify, or changes one or more existing extended fields in the profile of the partner you specify.

Input Parameters

partnerID	**String** The internal identifier of the trading partner for which to set extended fields.
fields	**Object List** The extended fields to set. Each extended field in the variable *fields* must be a com.wm.app.tn.profile.ExtendedProfileField instance.

Output Parameters

errors	**String List** (optional) A string array that contains the errors found in the extended fields, if any.

Usage Notes

The *Custom* and user-defined field groups contain extended fields that are not associated with any other group. See *webMethods Trading Networks Administrator's Guide* for a description of extended fields.

wm.tn.profile:setPartnerBinary

Adds, updates, or deletes a binary value for a trading partner. Supply either a binaryTypeCode or a binaryTypeDescription. If values for both parameters are supplied, the binaryTypeDescription is ignored.

Input Parameters

partnerID	**String** (optional) The internal identifier for the trading partner.
binary	**Byte [ ]** (optional) The binary values to be persisted to the ProfileStore. If the value is null, the service considers this as a delete operation.
binaryTypeCode	**String** (optional) A string that contains the numeric code for the type of binary to be set. This value must match the Type column of the PartnerBinary table. Supply a value for binaryTypeCode or for binaryTypeDescription, but not both.
binaryTypeDescription	**String** (optional) A string that describes the type of binary to be set. This value must match the Description column of the PartnerBinary

table. Supply a value for binaryTypeDescription or for binaryTypeCode, but not both.

Output Parameters

None.

wm.tn.profile:undeleteProfile

Undeletes a profile that was previously deleted from the trading network.

Input Parameters

partnerID	**String** The internal identifier for the trading partner to undelete.

Output Parameters

None.

wm.tn.profile:updateAddresses

Updates one or more addresses in a partner's profile. The service validates the addresses before saving them. If the service finds errors, it does not save the addresses and returns the errors.

Input Parameters

addresses	**Object List** The addresses that you want to update. The variable *addresses* must be an array of com.wm.app.tn.profile.Address instances.

Output Parameters

errors	**String List** (optional) A string array that contains the errors found in the address, if any.

wm.tn.profile:updateConnections

Updates one or more delivery method methods in a partner's profile. The service validates the connection information before saving it. If the service finds errors, it does not save the connection information and returns the errors.

Input Parameters

connections	**Object List** The delivery methods to update. The variable *connections* must be an array of com.wm.app.tn.profile.Destination instances.

Output Parameters

errors	**String List** (optional) A string array that contains errors found in the delivery methods, if any.

Usage Notes

You can associate, at most, one of each of the following types of com.wm.app.tn.profile.Destination instances with a partner:

- Primary HTTP
- Secondary HTTP
- Primary HTTPS
- Secondary HTTPS
- Primary SMTP
- Secondary SMTP
- Primary FTP
- Secondary FTP
- A delivery method that you created using HTTP, FTP, E-mail, and so on
- A custom delivery method that you created and registered using wm.tn.delivery:registerService

wm.tn.profile:updateContacts

Updates one or more contacts in a partner's profile. The service validates the contacts before saving them. If the service finds errors, it does not save the contacts and returns the errors.

Input Parameters

contacts	**Object List** The contacts to update. The variable *contacts* must be an array of com.wm.app.tn.profile.Contact instances.

Output Parameters

errors	**String List** (optional) A string array that contains the errors found in the contacts, if any.

wm.tn.profile:updateCorporation

Updates the corporate information for a partner in the trading network. The service validates the corporate information before saving it. If the service finds error, it does not save the corporate information and returns the errors.

Input Parameters

corporation	**Object** The corporate information to update. The *corporation* variable must be an instance of com.wm.app.tn.profile.Corporation.

Output Parameters

errors	**String List** (optional) A string array that contains the errors found in the corporate information, if any.

wm.tn.profile:updateIDs

Updates one or more external IDs in a partner's profile. The service validates the IDs before saving them. If the service finds errors, it does not save any IDs and returns the errors.

Input Parameters

ids	**Object List** The external IDs to update. The variable *ids* must be an array of com.wm.app.tn.profile.ID instances.

Output Parameters

errors	**String List** (optional) A string array that contains the errors found in the external IDs, if any.
username	**String** (optional) If Trading Networks created an Integration Server user account, this is the user name for the user account.
password	**String** (optional) If Trading Networks created an Integration Server user account, this is the password of the user account.

wm.tn.profile:updateProfileGroups

Updates one or more profile groups that a partner is a member of.

Input Parameters

partnerID	**String** The internal identifier for the trading partner.
profileGroupIds	**String List** A list of profile group IDs of which this partner is a member.

Output Parameters

None.

wm.tn.profile.create:newAddress

Creates an empty Address component for a profile.

Input Parameters

None.

Output Parameters

address	**Document** An uninitialized address; that is, all fields in the address are null. For the structure of *address*, see wm.tn.rec:Address. For Java developers, this is an instance of a com.wm.app.tn.profile.Address.

Usage Notes

- Use this service to create an empty address. You can use the address in the profile as either a corporate address or the address of a contact.
- After using this service to create an empty Address component, map or hardcode values into it before saving the address to the Trading Networks database.
  - If you are adding the address to a new profile, leave the internal ID fields (i.e., *AddressID*, *PartnerID*, and *ContactID*) empty. Trading Networks generates these internal IDs for you when you save the profile to the Trading Networks database.
  - If you are adding the address to an existing profile, you *must* specify an internal ID field to associate the new address with either the corporation or the contact within the existing profile. Specify *PartnerID* if you are adding the contact to the corporation. Specify *ContactID* if you are adding the address to an existing contact in the profile. Leave the internal ID, *AddressID*, empty; Trading Networks generates this internal ID for you.

wm.tn.profile.create:newContact

Creates an empty Contact component for a profile.

Input Parameters

None.

Output Parameters

contact	**Document** An uninitialized contact; that is, all fields in the contact are null. For the structure of *contact*, see wm.tn.rec:Contact. For Java developers, this is an instance of a com.wm.app.tn.profile.Contact.

Usage Notes

After using this service to create an empty Contact component, map or hardcode values into it before saving the contact to the Trading Networks database.

- If you are adding the contact to a new profile, leave the internal ID fields (*PartnerID* and *ContactID*) empty. Trading Networks generates these internal IDs for you when you save the profile to the Trading Networks database.
- If you are adding the contact to an existing profile, you *must* specify the internal ID field, *PartnerID*, to identify the profile to which to add the contact. Leave the internal ID, *ContactID* empty; Trading Networks generates this internal ID for you.

wm.tn.profile.create:newDelivery

Creates an empty delivery method component for a profile.

Input Parameters

None.

Output Parameters

delivery	**Document** An uninitialized delivery method component; that is, all fields in the delivery method are null. For the structure of *delivery*, see wm.tn.rec:Delivery. For Java developers, this is an instance of a com.wm.app.tn.profile.Destination.

Usage Notes

After using this service to create an empty delivery method component, map or hardcode values into it before saving the delivery method to the Trading Networks database.

- If you are adding the delivery method to a new profile, leave the internal ID fields (*PartnerID* and *DestinationID*) empty. Trading Networks generates these internal IDs for you when you save the profile to the Trading Networks database.
- If you are adding the delivery method to an existing profile, you *must* specify the internal ID field, *PartnerID*, to identify the profile to which to add the delivery method. Leave the internal ID, *DestinationID*, empty; Trading Networks generates this internal ID for you.

wm.tn.profile.create:newExtendedFields

Creates an extended field with no value.

Input Parameters

group **String** The group of extended fields to create. Valid values are:

- `Corporate` Creates all extended fields in the Corporate group.
- `Contact` Creates all extended fields in the Contact group.
- `Delivery` Creates all extended fields in the Delivery group.
- `Address` Creates all extended fields in the Address group.
- `ID` Creates all extended fields in the ID group.
- the name of another group - Specify the name of other groups that you previously created using the wm.tn.dictionary:addFieldGroup service. This service creates all extended fields for the group you specify.
- (null) Creates all extended fields for all groups (if you do *not* specify *group*).

Output Parameters

fields **Document List** A list of extended fields. The fields have no values. For the structure of each IS document type (IData object) in the *fields* IS document list, see wm.tn.rec:Field. For Java developers, this is an instance of a com.wm.app.tn.profile.ExtendedProfileField.

Usage Notes

- Each extended field returned by this service contains a fully initialized wm.tn.rec:FieldMetaData. You *must* specify the internal ID field, *PartnerID*, to identify the profile to which to add the extended field. Map or hard code a value into each *Value* field before saving the list of extended fields to the Trading Networks database.
- You cannot save extended fields with a new profile. You must save the profile first; then you can save the extended fields.

wm.tn.profile.create:newExternalID

Creates an empty external ID component for a profile.

Input Parameters

None.

Output Parameters

id	**Document** An uninitialized external ID, that is, all fields in the external ID are null. For the structure of *id*, see wm.tn.rec:ExternalID. For Java developers, this is an instance of a com.wm.app.tn.profile.ID.

Usage Notes

After using this service to create an empty external ID component, map or hardcode values into it before saving the external ID to the Trading Networks database.

- If you are adding the external ID to a new profile, leave the internal ID fields (*InternalID* and *PartnerIDID* empty. Trading Networks generates these internal IDs for you when you save the profile to the Trading Networks database.
- If you are adding the external ID to an existing profile, you *must* specify the internal ID field, *InternalID*, to identify the profile to which to add the external ID. Leave the internal ID, *PartnerIDID*, empty; Trading Networks generates this internal ID for you.

wm.tn.profile.create:newPrivateQueue

Creates a private queue for a trading partner.

Input Parameters

partnerID	**String** The internal ID of the partner that this private queue belongs to.

state **String** Possible values are `enabled`, `suspended`, `draining`, and `disabled`. For explanations of delivery queue states, see *webMethods Trading Networks Administrator's Guide*.

deliverySchedule **Document** Defines how and when tasks on this private queue are executed to deliver documents. The document contains the following keys:

- *svcName* **String** Name of the registered scheduled delivery service. This is not the fully-qualified name of the service in the Integration Server namespace; it is the name used to register the delivery service, using wm.tn.delivery:registerService. This is the name that is displayed in the delivery settings for the service in My webMethods.
- *svcInputs* **Document** Input variables to the scheduled delivery service. The variables you will supply are defined by the delivery service. If you are using the Batch FTP delivery service, see "wm.tn.transport:batchFtp" on page 290.
- *scheduleType* **String** Possible values are once, repeat, and complex. For explanations of delivery queue schedules, see *webMethods Trading Networks Administrator's Guide*.
- *oneTimeSchedule* **Document** Determines the date and time that the delivery service in svcName is invoked, when the value of *scheduleType* is once. If the value of *scheduleType* is once, *oneTimeSchedule* is required. This document contains the following keys:
  - *date* is a **String** in the format yyyy:mm:dd.
  - *time* is a **String** in the format hh:min:ss.
- *repeatingSchedule* **Document** Determines the interval at which the delivery service in *svcName* is invoked when the value of *scheduleType* is `repeat`. If *scheduleType* is `repeat`, *repeatingSchedule* is required.
  - *interval* **String** that indicates a number of seconds.
  - *noOverlap* **String** that indicates whether invocations of the delivery service may overlap. If the delivery service has not completed when it is invoked again, this setting determines whether to delay the invocation until the previous execution has completed. If *noOverlap* is `true`, Integration Server waits until the previous execution of the service has completed. If the value is `false`, it does not wait, making it possible for two separate threads of executions to deliver from the queue at the same time. The default is `false`.

complexSchedule **Document** Determines when the delivery service in *svcName* will be invoked when *scheduleType* is `complex`. If *scheduleType* is `complex`, *complexSchedule* is required. This document contains the following keys:

- *startDate* **String** The date when the delivery service will be invoked first, in the format yyyy:mm:dd.

- *startTime* **String** The time when the delivery service will be invoked first, in the format hh:mm:ss. If you do not specify a value, the server uses `00:00:00` (midnight).
- *endDate* **String** The date when the invocation of the delivery service will cease, in the format yyyy:mm:dd.
- *endTime* **String** The time when the invocation of the delivery service will cease, in the format hh:mm:ss. If you do not specify a value, the server uses `00:00:00` (midnight).
- *months* **String List** The months that the delivery service will be invoked. Values are `1` through `12`. If you do not specify a value, the service is invoked every month.
- *daysOfMonth* **String List** The days of the month that the delivery service will be invoked. Values are `1` through `31`. If you do not specify a value, the service is invoked every day of the month.
- *daysOfWeek* **String List** The days of the week that the delivery service will be invoked. Values are `1` (Sunday) through `31` (Saturday). If you do not specify a value, the service is invoked every day of the week.
- *hours* **String List** The hours of the day that the delivery service will be invoked. Values are `0` through `23`. If you do not specify a value, the service is invoked every hour.
- *minutes* **String List** The minutes within an hour that the delivery service will be invoked. Values are `0` through `59`. If you do not specify a value, the service is invoked every minute.

Output Parameters

queue	**Object** The DeliveryQueue object created and saved to the database. For Java programmers, this is com.wm.app.tn.delivery.DeliveryQueue.
msgs	**String List** This service invokes wm.tn.queuing:registerQueue. If that service returned any warning messages, they display here.

Usage Notes

This service validates the supplied inputs, create a DeliveryQueue object, saves it to the Trading Networks database, and updates the partner's profile to refer to the new private queue.

- If the queue was saved in an `enabled` or `draining` state, the specified delivery service will be scheduled for execution. If the queue was saved in an `enabled` or `suspended` state, the queue will be available to receive new delivery tasks immediately.
- After a private queue has been created with this service, you can view, update, or delete it using My webMethods.

- You cannot create a private queue for *Your Enterprise*, the profile that represents the owner (or "hub") of the trading network. Trading Networks does not deliver documents to *Your Enterprise*.

wm.tn.profile.create:newProfile

Creates a Profile with an empty Corporation, no Delivery Methods, no Contacts, no external IDs, and no Addresses.

Input Parameters

None.

Output Parameters

profile	**Document** An uninitialized profile; that is, all fields in the profile are null. For the structure of *profile*, see wm.tn.rec:Profile. For Java developers, this is an instance of a com.wm.app.tn.profile.Profile.

Usage Notes

- Use this service to create an empty profile. To create additional components for the profile (for example, Addresses, Contacts, Delivery Methods, External IDs), use the other wm.tn.profile:create services that are described in this section. After you create the profile and the additional components that you want, you can save the newly created profile to the Trading Networks database by invoking the wm.tn.profile:addProfile service and passing it the newly created profile.
- After creating the empty profile, map or hard code values into it before saving it to the Trading Networks database.When assigning values to the fields in a new profile, leave all the internal ID fields (for example, *PartnerID*, *ContactID*, *DestinationID*) empty. Trading Networks generates the internal IDs for you when you save the profile to the Trading Networks database.

14 Query Folder

Overview

Use query services (services in the wm.tn.query folder) to query the Trading Networks database for information about documents and activity log entries.

Summary of Elements in this Folder

The following table describes the elements that are available in this folder:

Element	Description
wm.tn.query:createDocumentQuery	Creates a query for documents saved in the Trading Networks database.
wm.tn.query:createEventQuery	Creates a query for activity log entries in the Trading Networks database.
wm.tn.query:createProfileQuery	Creates a query for partner profiles in the Trading Networks database.
wm.tn.query:createTaskQuery	Creates a query for delivery and service execution tasks.
wm.tn.query:createTPAQuery	Creates a TPA query object.
wm.tn.query:deliveryServiceDelete	Deletes a delivery service based on the query.
wm.tn.query:deliveryServiceQuery	Queries the Trading Networks database for delivery services.
wm.tn.query:documentQuery	Queries the Trading Networks database for documents.
wm.tn.query:doQuery	Executes a query based on the details you provide in the input parameters.
wm.tn.query:eventDelete	Deletes an activity log entry from Trading Networks database.
wm.tn.query:eventQuery	Queries the Trading Networks database for activity log entries.
wm.tn.query:getQueryResults	Retrieves the results of the query that you executed using the wm.tn.query.doQuery service.
wm.tn.query:getSQL	Retrieves the SQL query that you executed. You must have the 'Show SQL' functional permission to use this service.
wm.tn.query:profileQuery	Queries the Trading Networks database for partner profiles.
wm.tn.query:taskDelete	Deletes a task from Trading Networks database.
wm.tn.query:taskQuery	Queries the Trading Networks database for tasks.
wm.tn.query:tpaQuery	Queries the TPA store.

wm.tn.query:createDocumentQuery

Creates a query for documents saved in the Trading Networks database.

Input Parameters

senderId **String** (optional) The internal partner ID for sender of the documents to match.

receiverId **String** (optional) The internal partner ID for receiver of the documents to match.

messageTypeId **String** (optional) The internal TN document type ID of the documents to match.

routingStatus **String** (optional) The processing status of the documents to match.

userStatus **String** (optional) The user status of the documents to match.

documentId **String** (optional) The document ID for the documents to match.

internalDocId **String** (optional) The Trading Networks-generated internal ID of the document to match.

timeInterval **String** (optional) The time period in which the documents were received by Trading Networks. Specify one of the following:

- `TODAY`
- `YESTERDAY`
- `LAST_7_DAYS`
- `THIS_WEEK`
- `LAST_WEEK`
- `THIS_MONTH`
- `LAST_MONTH`
- `YTD`

attribs **Document List** (optional) The custom attribute criteria to use to match documents. For each custom attribute in the criteria, specify an IS document (IData object) in *attribs* with the following structure:

- *attribName*- Name of the attribute to use as search criteria. Use either *attribName* or *attribId* to identify the attribute.
- *attribId*- Internal ID of the attribute to use as search criteria. Use either *attribName* or *attribId* to identify the attribute.

- *op*- The operation to perform to match documents. Based on the data type of the attribute, specify the applicable operators. For the list of operators based on the data type, see "Usage Notes" on page 198.
- *attribValue*- The value to use when matching documents. The service uses this value and operation specified in *op* to perform the match.

  You do not need to specify a value for *attribValue* if you specify either `IS NULL` or `IS NOT NULL` for *op*.

  When specifying DATETIME values, use the format `yyyy-mm-dd hh:mm:ss.` *fff* where `.` *fff* represents nanoseconds. Specifying nanoseconds is optional.

  When specifying NUMBER values, use the format `xxx.` *xxx* where the fractional part is optional.

Output Parameters

query	**Object** An instance of com.wm.app.tn.db.ComplexDocQuery.

Usage Notes

For the input parameter *attribs*, in the *op* variable of the document, specify one of the following operators based on the data type of the attribute:

For this data type...	Specify one of the following...

STRING	`=`
	`IS NULL`
	`IS NOT NULL`
	`CONTAINS`

NUMBER	`=`
	`<>`
	`>=`
	`<=`
	`<`
	`>`
	`IS NULL`
	`IS NOT NULL`

For this data type...	Specify one of the following...
DATETIME	= IS NULL IS NOT NULL BEFORE AFTER
STRING LIST	IS NULL IS NOT NULL INCLUDES
NUMBER LIST	IS NULL IS NOT NULL INCLUDES
DATETIME LIST	IS NULL IS NOT NULL

wm.tn.query:createEventQuery

Creates a query for activity log entries in the Trading Networks database.

Input Parameters

entryType **String** (optional) The type of entries to match. Specify `ERROR`, `WARNING`, or `MESSAGE`.

entryClass **String** (optional) The category (or activity class) of entries to match. You can specify a value that you use when adding entries to the activity log or one of the following activity classes that Trading Networks sets:

- `Delivery`
- `Envelope`
- `Persistence`
- `Recognition`
- `Processing`
- `Validation`

- `Verification`
- `General`

For a description of the activity classes that Trading Networks uses, see information about using the activity log in the *webMethods Trading Networks User's Guide*.

internalDocId **String** (optional) The internal document ID for documents that are related to the activity log entries to match.

internalPartnerId **String** (optional) The internal partner ID for the trading partners that are related to the activity log entries to match.

conversationID **String** (optional) The conversation ID that is related to the activity log entries to match.

stepId **String** (optional) The conversation step ID that is related to the activity log entries to match.

B2BUser **String** (optional) The user name for the current user account when the activity log entries to match were added.

messageText **String** (optional) The message text (in either the activity log brief or full message) for the activity log entries to match.

timeInterval **String** (optional) The time period in which the activity log entries were created by Trading Networks. Specify one of the following:

- `TODAY`
- `YESTERDAY`
- `LAST_7_DAYS`
- `THIS_WEEK`
- `LAST_WEEK`
- `THIS_MONTH`
- `LAST_MONTH`
- `YTD`

Output Parameters

query **Object** An instance of com.wm.app.tn.db.EventQuery.

wm.tn.query:createProfileQuery

Creates a query for partner profiles in the Trading Networks database.

Input Parameters

corpName **String** (optional) The corporation name of the trading partner(s).

unitName **String** (optional) The unit name of the trading partner(s).

status **String** (optional) The status of the trading partner(s). Valid values are:

- `Active` To match only active partner profiles.
- `Inactive` To match only inactive partner profiles.

For more information, see wm.tn.profile:changeStatus.

externalIDType **String** (optional) The external ID type (e.g. `DUNS`). You can query only one external ID type with this service.

externalIDValue **String** (optional) The value for the specified external ID type.

groupName **String** (optional) The group name of the trading partner(s).

city **String** (optional) The city in the address.

state **String** (optional) The state or province for the address.

zip **String** (optional) The zip code or postal code for the address.

country **String** (optional) The country for the address.

extendedFields **Document List** (optional) The extended fields criteria to match. For each extended field, specify an IS document (IData object) in this parameter with the following structure:

- *fieldId* - The internal ID that uniquely identifies the extended fields to use in the criteria. You can query only extended fields of type String.
- *op* - The operation to use in matching partner profiles. For a String data type, select one of the following:
  - `=`
  - `<>`
  - `IS NULL`
  - `IS NOT NULL`
  - `CONTAINS`
- *fieldValue* - **String** The extended fields to match. This service uses this parameter together with the *op* parameter to perform the match.

  Note:
  If you specify `IS NULL` or `IS NOT NULL`, *fieldValue* is not required.

username **String** (optional) User name of the user mapped to the profiles to retrieve.

Output Parameters

query **Object** An instance of com.wm.app.tn.db.EventQuery.

wm.tn.query:createTaskQuery

Creates a query for delivery and service execution tasks.

Input Parameters

internalDocId **String** (optional) The internal document ID of the tasks to match.

taskId **String** (optional) The ID of the task to match.

taskType **String** (optional) The task type of the tasks to match. Valid values are:

- `Delivery` - To match only delivery tasks.
- `Service Execution` - To match only service execution tasks.

internalPartnerId **String** (optional) The internal partner ID of the partner that is associated with the tasks to match.

taskStatus **String** (optional) The status of the tasks to match. Valid values:

- NEW
- PENDING
- DONE
- FAILED
- STOPPED

deliveryMethod **String** (optional) The delivery method of the tasks to match. Use wm.tn.delivery:getRegisteredServices to get a list of registered delivery services. You can specify the name of any registered service for *deliveryMethod*.

serverId **String** (optional) The server ID of the server that is processing the tasks to match.

timeCreated **String** (optional) The time period in which Trading Networks created the tasks. Specify one of the following:

- `TODAY`
- `YESTERDAY`

- `LAST_7_DAYS`
- `THIS_WEEK`
- `LAST_WEEK`
- `THIS_MONTH`
- `LAST_MONTH`
- `YTD`

queueName **String** The name of the delivery queue to match tasks.

Output Parameters

query **Object** An instance of com.wm.app.tn.db.DeliveryJobQuery.

wm.tn.query:createTPAQuery

Creates a TPA query object. If the TPA is not found, the service reports an error. For other service invocation- or DB-related errors, it throws an exception.

Input Parameters

senderID **String** (optional) The ID of the trading partner that has the sender role in the transaction the TPA governs.

receiverID **String** (optional) The ID of the trading partner that has the receiver role in the transaction the TPA governs.

agreementID **String** The agreement ID of the TPA.

dataSchema **String** (optional) A blueprint of the TPA that establishes the TPA parameters and values.

status **String** (optional) The status of the TPA. It can have one of three values: `proposed`, `disabled`, or `agreed`.

exportService **String** (optional) The fully-qualified name of a service that exports a Trading Networks TPA and converts it to an industry-standard format.

initService **String** (optional) The fully-qualified name of a service that sets default values for the IS document type defined by *dataSchema*.

timeInterval **String** (optional) The time period in which the activity log entries were created by Trading Networks. Specify one of the following:

- `TODAY`

- YESTERDAY
- LAST_7_DAYS
- THIS_WEEK
- LAST_WEEK
- THIS_MONTH
- LAST_MONTH
- YTD

Output Parameters

query **Object** The query object.

wm.tn.query:deliveryServiceDelete

Deletes a delivery service based on the query.

Input Parameters

query **Object** The query to run. The query must be an instance of com.wm.app.tn.db.SimpleDocQuery. You can use wm.tn.query:createDocumentQuery to create this query object.

Output Parameters

resultCount **String** The number of rows in the query result.

wm.tn.query:deliveryServiceQuery

Queries the Trading Networks database for delivery services.

Input Parameters

query **Object** The query to run. The query must be an instance of com.wm.app.tn.db.SimpleDocQuery. You can use wm.tn.query:createDocumentQuery to create this query object.

disablePaging **String** (optional) Whether to return the results of the service or an enumeration ID that you can use to get the results a page at a time. Valid values are:

- `true` - Returns the results of the query.
- `false` - Default. Returns an enumeration ID.

  Use the enumeration ID as input into the services in the wm.tn.enumerate services to get the results a page at a time. For more information, see "Enumerate Folder" on page 119.

pageSize **String** The page size to use when enumerating over the query results. This variable is only used when *disablePaging* is `false`. The default is 25.

maxRowCount **String** (optional) The maximum number of rows of results to return. The service silently drops excess rows. Specify `0` to return all results. The default is `0`.

queryTimeout **String** (optional) Ignored.

threshold **String** (optional) The number of rows of query results to store in the session object to optimize query execution. The service stores the remaining rows in the Integration Server repository. For best performance, specify a value equal to the page size. If you do not specify a value, the service uses -1 causing the service to use the value specified by the `tn.query.threshold` property.

For more information about this property, view the online help files as follows: from Integration Server Administrator, click **Trading Networks** from the Solutions menu of the Navigation panel. Trading Networks displays the TN Properties page. In the upper right corner of the TN Properties page, click **Help**.

If you are using Trading Networks in a cluster, do not specify *threshold* and ensure the `tn.query.threshold` property is set to `-1`, which disables using the session object.

id **String** (optional) The unique identifier to store the paged query results. This parameter is applicable only when the results are paged. If this parameter is not set, an *id* is automatically generated.

Output Parameters

resultCount **String** The number of rows in the query result.

results **Document List** (optional) The results of the query.

id **String** (optional) The repository path.

wm.tn.query:documentQuery

Queries the Trading Networks database for documents.

Input Parameters

query **Object** The query to run. The query must be an instance of com.wm.app.tn.db.SimpleDocQuery. You can use wm.tn.query:createDocumentQuery to create this query object.

aggregate **String** (optional) Whether to aggregate query results. The aggregate version lists counts of all the documents satisfying the query grouped by sender and receiver. Valid values are:

- `true` Returns the aggregate version.
- `false` Default. Returns detailed information about each document.

disablePaging **String** (optional) Whether to return the results of the service or an enumeration ID that you can use to get the results a page at a time. Valid values are:

- `true` - Returns the results of the query.
- `false` - Default. Returns an enumeration ID.

  Use the enumeration ID as input into the services in the wm.tn.enumerate services to get the results a page at a time. For more information, see "Enumerate Folder" on page 119.

pageSize **String** (optional) The page size to use when enumerating over the query results. This variable is only used when *disablePaging* is `false`. The default is 25.

maxRowCount **String** (optional) The maximum number of rows of results to return. The service silently drops excess rows. Specify `0` to return all results. The default is `0`.

queryTimeout **String** (optional) Ignored.

threshold **String** (optional) The number of rows of query results to store in the session object to optimize query execution. The service stores the remaining rows in the Integration Server repository. For best performance, specify a value equal to the page size. If you do not specify a value, the service uses -1 causing the service to use the value specified by the `tn.query.threshold` property.

For more information about this property, view the online help files as follows: from Integration Server Administrator, click **Trading Networks** from the Solutions menu of the Navigation panel. Trading Networks

displays the TN Properties page. In the upper right corner of the TN Properties page, click **Help**.

If you are using Trading Networks in a cluster, do not specify *threshold* and ensure the `tn.query.threshold` property is set to `-1`, which disables using the session object.

id — **String** (optional) The unique identifier to store the paged query results. This parameter is applicable only when the results are paged. If this parameter is not set, an *id* is automatically generated.

Output Parameters

resultCount — **String** The number of rows in the query result.

results — **Document List** (optional) If *disablePaging* is `true`, this contains the results of the query. Each returned row is represented as an IS document (IData object) in *results*. The keys in each IS document depend on the parameters of the query.

id — **String** (optional) If *disablePaging* is `false`, this is an enumeration ID for use with the services in the wm.tn.enumerate folder.

wm.tn.query:doQuery

Executes a query based on the details you provide in the input parameters. After executing the query, you can execute the wm.tn.query:getQueryResults service to get the resulting data of the query.

Input Parameters

queryInput — **Document** The filter criteria for the query, the sort order, and the column names that must be returned in the result set. For the structure of *queryInput*, see wm.tn.rec:queryInput.

queryType — **String** The query type. Use one of the following values based on the query type:

- `1` - Document Query. Creates a query for documents saved in the Trading Networks database.
- `2` - Event Query. Queries the Trading Networks database for activity log entries.
- `3` - Task Query. Creates a query for delivery and service execution tasks.
- `4` - Profile Query. Creates a query for partner profiles in the Trading Networks database.

locale	**String** Optional. The locale for formatting the query results. Specify the two-letter ISO 639 language code for the language to use, and the two-letter ISO 3166 country code for the country that is associated with the language. Use the format: *<languagecode>_<countrycode>* For example, to localize for US English, set the locale as `'en_US'`.
queryIdToCancel	**String** Optional. The ID of a previously executed query to terminate. If the query is still executing, then the service terminates the query execution, immediately. If the query is done executing, then the service deletes the results from the service cache, immediately.
IsMWS	**String** Optional. Defines whether the service is called from My webMethods Server or from a user using Designer. Valid values are: ■ `yes` - Default. My webMethods Server calls the service. ■ `no` - A user using Designer calls the service.

Output Parameters

queryOutput	**Document** The output details of the executed query. For the structure of *queryOutput*, see wm.tn.rec:queryOutput.
svcResponse	**Document** Conditional. The error messages, warnings, and other information that the service generated. For the structure of *svcResponse*, see wm.tn.rec:svcResponse.

Usage Notes

The criteria for Extended Fields of profile query type should use FieldGroupName:FieldName in the fieldName (queryInput > criteria > fields > fieldName) field.

wm.tn.query:eventDelete

Deletes an activity log entry from Trading Networks database.

Input Parameters

query	**Object** The query to run. The query must be an instance of com.wm.app.tn.db.EventQuery. You can use wm.tn.query:createEventQuery to create this query object.

Output Parameters

resultCount	**String** The number of rows in the query result.

wm.tn.query:eventQuery

Queries the Trading Networks database for activity log entries.

Input Parameters

query	**Object** The query the service should run. The query must be an instance of com.wm.app.tn.db.EventQuery. You can use wm.tn.query:createEventQuery to create this query object.
pageSize	**String** (optional) The page size to use when enumerating over the query results. The default is 25.
maxRowCount	**String** (optional) The maximum number of rows of results to return. The service silently drops excess rows. Specify `0` to return all results. The default is `0`.
queryTimeout	**String** (optional) Ignored
threshold	**String** (optional) The number of rows of query results store in the session object to optimize query execution. The service stores the remaining rows in the Integration Server repository. For best performance, specify a value equal to the page size. If no value is specified, the service uses -1 causing the service to use the value specified by the `tn.query.threshold` property. For more information about this property, view the online help files as follows: From Integration Server Administrator, click **Trading Networks** from the Solutions menu of the Navigation panel. Trading Networks displays the TN Properties page. In the upper right corner of the TN Properties page, click **Help**. If you are using Trading Networks in a cluster, do not specify threshold and ensure the`tn.query.threshold` property is set to `-1`, which disables using the session object.
id	**String** (optional) The unique identifier to store the paged query results. This parameter is applicable only when the results are paged. If this parameter is not set, an *id* is automatically generated.

Output Parameters

resultCount	**String** The number of rows in the query result.

id **String** The enumeration ID for use with the services in the wm.tn.enumerate folder. For more information, see "Enumerate Folder" on page 119.

wm.tn.query:getQueryResults

Retrieves the results of the query that you executed using the wm.tn.query.doQuery service.

Input Parameters

queryID **String** The ID of the executed query for which results must be retrieved. The ID must be the same as that of the query that was executed using the wm.tn.query:doQuery service. The ID is available in the *queryOutput* parameter of the wm.tn.query:doQuery service. For more information, see wm.tn.query:doQuery.

pageNumber **String** The page number of the result set from which to retrieve the results. The query results are paginated based on the page size you specified in the wm.tn.query:doQuery service.

locale **String** Optional. The locale for formatting the query results. Specify the two-letter ISO 639 language code for the language to use, and the two-letter ISO 3166 country code for the country that is associated with the language.

Use the format: <*languagecode*>_<*countrycode*>

For example, to localize for U.S. English, set the locale as 'en_US'.

IsMWS **String** Optional. Defines whether the service is called from My webMethods Server or from a user using Designer. Valid values:

- yes Default. My webMethods Server calls the service.
- no A user calls the service using Designer.

Output Parameters

page **Document List** The results of the query. Each entry in the document list corresponds to a row in the result set. The document has the following fields:

- *rowData* **String List** The data that results after the query execution. Each entry in the list corresponds to a column in the result set. The number of columns in the list is same as the number of columns in the result set.
- *columns* **String List** The names of the columns in the result set. Each row in the list corresponds to a column name in the result set.

threadRunning **String** Indicates whether the thread executing the wm.tn.query:doQuery service is still running. Valid values are:

- `true` - Thread is running.
- `false` - Thread is not running.

rowsread **String** The total number of records in the result set.

svcResponse **Document** Conditional. The error messages, warnings, and other information that resulted while executing the service. For the structure of *svcResponse*, see wm.tn.rec:svcResponse.

wm.tn.query:getSQL

Retrieves the SQL query that you executed. You must have the 'Show SQL' functional permission to use this service. For more information about functional permissions and configuring My webMethods to work with Trading Networks, see *webMethods Trading Networks Administrator's Guide*.

Input Parameters

queryInput **Document** The filter criteria for the query, the sort order, and the column names that must be returned in the result set. For the structure of *queryInput*, see wm.tn.rec:queryInput.

queryType **String** The type of query. Use one of the following values based on the query type:

- `1` - Document Query. Creates a query for documents saved in the Trading Networks database.
- `2` - Event Query. Queries the Trading Networks database for activity log entries.
- `3` - Task Query. Creates a query for delivery and service execution tasks.
- `4` - Profile Query. Creates a query for partner profiles in the Trading Networks database.

locale **String** Optional. The locale for formatting the query results. Specify the two-letter ISO 639 language code for the language to use, and the two-letter ISO 3166 country code for the country that is associated with the language.

Use the format: *<languagecode>_<countrycode>*

For example, to localize for U.S. English, set the locale as '`en_US`'.

IsMWS **String** Optional. Defines whether the service is called from My webMethods Server or from a user using Designer. Valid values are:

- `yes` Default. Call the service from My webMethods Server.
- `no` Call the service using Designer.

Output Parameters

sql	**String**. The retrieved SQL query.
svcResponse	**Document** Conditional. The error messages, if any, that resulted while retrieving the SQL. For the structure of *svcResponse*, see wm.tn.rec:svcResponse.

wm.tn.query:profileQuery

Queries the Trading Networks database for partner profiles.

Input Parameters

query	**Object** The query that the service should run.The query must be an instance of com.wm.app.tn.db.EventQuery. You can use wm.tn.query:createProfileQuery to create this query object.
disablePaging	**String** (optional) Whether to return the results of the service or an enumeration ID that you can use to get the results a page at a time. Valid values are: - `true` Returns the results of the query. - `false` Default. Returns an enumeration ID. Use the enumeration ID as input into the services in the wm.tn.enumerate services to get the results a page at a time. For more information, see "Enumerate Folder" on page 119.
pageSize	**String** (optional) The page size to use when enumerating over the query results. The default is 25.
maxRowCount	**String** (optional) The maximum number of rows of results to return. The service silently drops excess rows. Specify `0` to return all results. The default is `0`.
queryTimeout	**String** (optional) Ignored.
threshold	**String** (optional) The number of rows of query results to store in the session object to optimize query execution. The service stores the remaining rows in the Integration Server repository. For best performance, specify a value equal to the page size. If you do not specify a value, the service uses -1 causing the service to use the value specified by the `tn.query.threshold` property.

For more information about this property, view the online help files as follows: From Integration Server Administrator, click **Trading Networks** from the Solutions menu of the Navigation panel. Trading Networks displays the TN Properties page. In the upper right corner of the TN Properties page, click **Help**.

If you are using Trading Networks in a cluster, do not specify *threshold* and ensure the `tn.query.threshold property` is set to `-1`, which disables using the session object.

id — **String** (optional) The unique identifier to store the paged query results. This parameter is applicable only when the results are paged. If this parameter is not set, an *id* is automatically generated.

Output Parameters

resultCount — **String** The number of rows in the query result.

id — **String** (optional) If *disablePaging* is `false`, this is an enumeration ID for use with the services in the wm.tn.enumerate folder.

wm.tn.query:taskDelete

Deletes a task from Trading Networks database.

Input Parameters

query — **Object** The query to run. The query must be an instance of com.wm.app.tn.db.DeliveryJobQuery. You can use wm.tn.query:createTaskQuery to create this query object.

Output Parameters

resultCount — **String** The number of rows in the query result.

wm.tn.query:taskQuery

Queries the Trading Networks database for tasks.

Input Parameters

query	**Object** The query that the service runs. The query must be an instance of com.wm.app.tn.db.DeliveryJobQuery. You can use wm.tn.query:createTaskQuery to create this query object.
disablePaging	**String** (optional) Whether the service returns the results of the service or an enumeration ID that you can use to get the results a page at a time. Valid values are: ■ `true` - Returns the results of the query. ■ `false` - Default. Returns an enumeration ID. Use the enumeration ID as input into the services in the wm.tn.enumerate services to get the results a page at a time. For more information, see "Enumerate Folder" on page 119.
pageSize	**String** (optional) The page size to use when enumerating over the query results. This variable is only used when *disablePaging* is `false`. The default is 25.
maxRowCount	**String** (optional) The maximum number of rows of results to return. The service silently drops excess rows. Specify `0` to return all results. The default is `0`.
queryTimeout	**String** (optional) Ignored.
threshold	**String** (optional) The number of rows of query results to store in the session object to optimize query execution. The service stores the remaining rows in the Integration Server repository. For best performance, specify a value equal to the page size. If no value is specified, the service uses -1, causing the service to use the value specified by the `tn.query.threshold` property. For more information about this property, view the online help files as follows: from Integration Server Administrator, click **Trading Networks** from the Solutions menu of the Navigation panel. Trading Networks displays the TN Properties page. In the upper right corner of the TN Properties page, click **Help**. If you are using Trading Networks in a cluster, do not specify *threshold* and ensure the `tn.query.threshold property` is set to `-1`, which disables using the session object.
id	**String** (optional) The unique identifier to store the paged query results. This parameter is applicable only when the results are paged. If this parameter is not set, an *id* is automatically generated.

Output Parameters

resultCount	**String** The number of rows in the query result.
results	**Document List** (optional) If *disablePaging* is `true`, this includes the results of the query. Each returned row is represented as an IS document (IData object) in *results*. The keys in each IS document depend on the parameters of the query.
id	**String** (optional) If *disablePaging* is `false`, this is an enumeration ID for use with the services in the wm.tn.enumerate folder.

wm.tn.query:tpaQuery

Queries the TPA store. If no TPA is found, the service returns null. For other service invocation- or DB-related errors, it throws an exception.

Input Parameters

query	**Object** The TPA query object.
disablePaging	**String** (optional) Determines whether the service returns the results of the service or an enumeration ID, which can be used to get the results a page at a time. Valid values are: ■ `true` - Returns the results of the query. ■ `false` - Default. Returns an enumeration ID. Use the enumeration ID as input into the services in the wm.tn.enumerate services to get the results a page at a time. For more information, see "Enumerate Folder" on page 119.
pageSize	**String** The page size to use when enumerating over the query results. This variable is only used when *disablePaging* is `false`. The default is `25`.
maxRowCount	**String** (optional) The maximum number of rows of results to return.
queryTimeout	**String** (optional) Ignored.
threshold	**String** (optional) The number of rows of query results to store in the session object to optimize query execution. The service stores the remaining rows in the Integration Server repository. For best performance, specify a value equal to the page size. If you do not specify a value, the service uses `-1`, causing the service to use the value specified by the `tn.query.threshold` property. For more information about this property, view the online help files as follows: From Integration Server Administrator, click **Trading Networks**

from the Solutions menu of the Navigation panel. Trading Networks displays the TN Properties page. In the upper right corner of the TN Properties page, click **Help**.

If you are using Trading Networks in a cluster, do not specify threshold and ensure the `tn.query.threshold` property is set to `-1`, which disables using the session object.

id **String** (optional) The unique identifier to store the paged query results. This parameter is applicable only when the results are paged. If this parameter is not set, an *id* is automatically generated.

Output Parameters

results **Document List** (optional) The results of the query.

resultCount **String** (optional) The number of rows in the query result.

15 Queuing Folder

Overview

Use the queuing services (services in the wm.tn.queuing folder) to maintain delivery queues and their delivery schedules. This folder also contains services that Trading Networks uses to deliver documents from a queue.

Several of the queuing services require the queue name as an input parameter. For a public queue, you specify the name given to the queue when it was defined. If you want to specify a private queue, the name of a private queue is the internal ID of the partner to which you are sending documents. The following lists ways you can obtain the receiver's internal ID based on the data that is in the pipeline:

- If the receiver's profile is in the pipeline, the internal ID is the *partnerID* variable that is within the *Corporate* variable of the receiver's profile. For the structure of the profile, see wm.tn.rec:Profile. For the structure of the *Corporate* variable, see wm.tn.rec:Corporation.
- If the document being delivered is in the pipeline, the receiver's internal ID is the ReceiverID variable that is within the *bizdoc* variable. For the structure of *bizdoc* see wm.tn.rec:BizDocEnvelope.
- If one of the receiver's external IDs, such as a D-U-N-S number, is in the pipeline, you can invoke the wm.tn.profile:getInternalID to get the receiver's internal ID.

Summary of Elements in this Folder

The following table describes the elements that are available in this folder:

Element	Description
wm.tn.queuing:deliverBatch	Attempts to deliver the documents associated with all delivery tasks in the specified queue.
wm.tn.queuing:getQueuedTask	Dequeues a delivery task from a scheduled delivery queue.
wm.tn.queuing:getRegisteredQueue	Retrieves information about a specified schedule delivery queue.
wm.tn.queuing:getRegisteredQueues	Retrieves a list of the scheduled delivery queues that are registered with Trading Networks.
wm.tn.queuing:listQueuedTasks	Returns the list of delivery task IDs of all delivery tasks that are in a specified delivery queue.
wm.tn.queuing:queueDocument	Schedules a document for delivery by creating a delivery task for the document, setting the name of the scheduled delivery queue in which it places the delivery task, and setting the status of the delivery task to QUEUED.
wm.tn.queuing:reassign	Reassigns the delivery tasks that are in a scheduled delivery queue to another scheduled delivery queue.

Element	Description
wm.tn.queuing:registerQueue	Adds a new delivery queue to the Trading Networks database and makes it available for use.
wm.tn.queuing:removeQueue	Deletes a scheduled delivery queue.
wm.tn.queuing:updateQueue	Updates an existing scheduled delivery queue in the Trading Networks database and modifies the queue's delivery schedule as appropriate.
wm.tn.queuing:updateQueuedTask	Updates the delivery status of a delivery task in a scheduled delivery queue.

wm.tn.queuing:deliverBatch

Attempts to deliver the documents associated with all delivery tasks in the specified queue.

Input Parameters

queue **String** The name of the scheduled delivery queue.

Output Parameters

None

Usage Notes

When you define a scheduled delivery queue, Trading Networks schedules this service to run on the Integration Server corresponding to the delivery schedule that you specify. If you want to deliver documents outside of the scheduled run times, you can invoke this service directly.

wm.tn.queuing:getQueuedTask

Dequeues a delivery task from a scheduled delivery queue.

Input Parameters

queue **String** The name of the scheduled delivery queue from which you want to dequeue a delivery task.

taskId **String** (optional) The task ID of the task to be dequeued from the scheduled delivery queue.

Output Parameters

task	**Document** The task in the scheduled delivery queue specified by *taskId*. If you do not specify the task ID, the oldest task in the scheduled delivery task specified by *queue*. For the structure of *task*, see wm.tn.rec:Task.
timeDequeued	**Object** A timestamp indicating when this service dequeued the delivery task returned in *task*. For Java developers, this is an instance of java.lang.Long.

Usage Notes

- Use this service when you create a scheduled delivery service. Typically, the scheduled delivery service uses this service to dequeue a delivery task and then delivers the document. After delivering the document, your scheduled delivery service should invoke wm.tn.queuing:updateQueuedTask to update the status of the queued delivery task. Note that wm.tn.queuing:getQueuedTask may throw an OutOfMemoryError if a scheduled delivery service has a very large output pipeline. To avoid this error, place the pipeline data from the scheduled delivery service into the *serviceOutput* parameter of wm.tn.queuing:updateQueuedTask.

  For an example of a scheduled delivery service, see the wm.tn.transport:batchFtp service. For a description about how to create a scheduled delivery service, including how to use the wm.tn.queuing:getQueuedTask service, see *webMethods Trading Networks Administrator's Guide*.
- The wm.tn.queuing:updateQueuedTask service uses the *timeDequeued* value. Do *not* modify this value or drop it from the pipeline before invoking wm.tn.queuing:updateQueuedTask.
- If you do not specify a *taskId*, Trading Networks dequeues the oldest updated task in the queue (first in, first out).

wm.tn.queuing:getRegisteredQueue

Retrieves information about a specified schedule delivery queue.

Input Parameters

name	**String** The name of the scheduled delivery queue to retrieve.

Output Parameters

queue	**Object** The requested scheduled delivery queue. This is a DeliveryQueue object, an instance of com.wm.app.tn.delivery.DeliveryQueue.

Usage Notes

If Trading Networks does not have a registered delivery queue with the name specified in *name*, *queue* is null.

wm.tn.queuing:getRegisteredQueues

Retrieves a list of the scheduled delivery queues that are registered with Trading Networks.

Input Parameters

includePrivate	**String** Whether you want the service to return information about private scheduled delivery queues. Valid values are: ■ `false` - Default. Retrieves only public scheduled delivery queues. ■ `true` - Retrieves both public and private scheduled delivery queues.

Output Parameters

queues	**Object List** A list of registered scheduled delivery queues. This is an array of DeliveryQueue objects, instances of com.wm.app.tn.delivery.DeliveryQueue. Each queue is returned as a DeliveryQueue object.

Usage Notes

If Trading Networks has no registered delivery queues, *queues* is null.

wm.tn.queuing:listQueuedTasks

Returns the list of delivery task IDs of all delivery tasks that are in a specified delivery queue.

Input Parameters

queue	**String** The name of the scheduled delivery queue for which you want a list of delivery tasks.

Output Parameters

taskids	**String List** A list of delivery task IDs, one for each delivery task in the scheduled delivery queue.

Usage Notes

To retrieve the details for a specific delivery task, invoke the wm.tn.task:getTask service and pass it a task ID returned by this service.

wm.tn.queuing:queueDocument

Schedules a document for delivery by creating a delivery task for the document, setting the name of the scheduled delivery queue in which it places the delivery task, and setting the status of the delivery task to `QUEUED`.

Input Parameters

bizdoc	**Object** The document to schedule for delivery. Trading Networks creates a delivery task for the document and places the delivery task in queue specified by *queue*. For Java developers, this is an instance of com.wm.app.tn.doc.BizDocEnvelope.
queue	**String** The name of the scheduled delivery queue to use for delivery of the document.

Output Parameters

None.

Usage Notes

You can use My webMethods to create processing rules that schedule documents for delivery. Use this service to programmatically schedule documents for delivery.

wm.tn.queuing:reassign

Reassigns the delivery tasks that are in a scheduled delivery queue to another scheduled delivery queue. The status, retry count, and retry limit associated with a reassigned delivery tasks are not affected by this operation. This service reassigns the delivery tasks, but does not change them in any other way.

Input Parameters

from	**String** The name of the scheduled delivery queue that contains the delivery tasks to reassign to another queue.
to	**String** The name of the scheduled delivery queue to which you want to assign the delivery tasks.

Output Parameters

count	**String** The number of delivery tasks that the service reassigned.

wm.tn.queuing:registerQueue

Adds a new delivery queue to the Trading Networks database and makes it available for use.

Input Parameters

queue	**Object** The scheduled delivery queue to add. It must be a DeliveryQueue object, an instance of com.wm.app.tn.delivery.DeliveryQueue. It cannot have the same name as an existing queue and it must contain a valid DeliverySchedule, an instance of com.wm.app.tn.delivery.DeliverySchedule.

Output Parameters

msgs	**String List** (optional) If this service encountered errors while adding the queue, *msgs* contains descriptions of the errors.

Usage Notes

If the queue is added in an enabled or draining state, this service creates a scheduled task in the Integration Server so documents added to the queues are delivered at the times defined by the delivery schedule. For more information about tasks in Integration Server, see *webMethods Integration Server Administrator's Guide*.

wm.tn.queuing:removeQueue

Deletes a scheduled delivery queue. After the schedule delivery queue is deleted, Trading Networks no longer displays the queue in My webMethods and you can no longer use it to deliver documents. When you execute this service, the service deletes scheduled delivery queue from the Trading Networks database and cancels the associated scheduled task in Integration Server.

Input Parameters

name	**String** The name of the scheduled delivery queue to delete.

Output Parameters

None.

Usage Notes

If this service encounters a problem while deleting the scheduled delivery queue, it throws a ServiceException.

wm.tn.queuing:updateQueue

Updates an existing scheduled delivery queue in the Trading Networks database and modifies the queue's delivery schedule as appropriate.

Input Parameters

queue	**Object** The scheduled delivery queue to update. It must be a DeliveryQueue object, an instance of com.wm.app.tn.delivery.DeliveryQueue. The DeliveryQueue object must identify a queue that is registered with Trading Networks; that is, the name of the queue must match the name of a registered scheduled delivery queue. Additionally, the DeliveryQueue object must contain a valid DeliverySchedule, an instance of com.wm.app.tn.delivery.DeliverySchedule.

Output Parameters

msgs	**String List** (optional) If this service encountered errors while updating the queue, *msgs* contains descriptions of the errors.

Usage Notes

If the update to the queue changes the queue state (enabled, disabled, draining, or stopped), Trading Networks performs the following against the scheduled task in the Integration Server:

Original queue state	Updated queue state	The task in the Integration Server is…
disabled or stopped	enabled or draining	■ Resumed if the task exists ■ Created if not task exists
enabled or draining	disabled or stopped	■ Suspended
disabled or stopped	disabled or stopped	■ Unchanged
enabled or draining	enabled or draining	■ Updated with new run dates and times as specified

wm.tn.queuing:updateQueuedTask

Updates the delivery status of a delivery task in a scheduled delivery queue.

Input Parameters

taskid	**String** Internal identifier of the delivery task to update.
queue	**String** The name of the scheduled delivery queue in which the delivery task you want to update resides.
status	**String** The status to assign to the delivery task. Valid values are: ■ `success` The attempt to deliver the document associated with the delivery task was successful. Trading Networks updates the task status to DONE and the document status to DONE. ■ `fail` The attempt to deliver the document associated with the delivery task failed. If the delivery task has not reached the retry limit, Trading Networks increments the retry count and sets the task status to QUEUED. If the delivery task reaches the retry limit, Trading Networks updates the task status to FAILED and the document status to FAILED.
statusMsg	**String** (optional) The message that the transport service returns after attempting to deliver the document.
timeDequeued	**Object** (optional) A timestamp indicating when the task was dequeued using the wm.tn.queuing:getQueuedTask service. The wm.tn.queuing:getQueuedTask service placed this value into the pipeline. Trading Networks uses this value to determine how long it took to deliver the document associated with the delivery task. For Java developers, this is an instance of java.lang.Long.
serviceOutput	**Document** (optional) Data to save to the database. Specify the data as key/value pairs. The data type of *serviceOutput* is com.wm.data.IData. This is useful when a scheduled delivery service has a very large output pipeline. When you use wm.tn.queuing:getQueuedTask to dequeue a delivery task from a scheduled delivery queue, wm.tn.queuing:getQueuedTask might throw an OutOfMemoryError if the scheduled delivery service has a very large output pipeline. To avoid this error, place the pipeline data from the scheduled delivery service into the *serviceOutput* parameter of wm.tn.queuing:updateQueuedTask. Data in the *serviceOutput* parameter is saved to the database.

Output Parameters

None.

Usage Notes

- Use this service when you create a scheduled delivery service. Typically, the scheduled delivery service invokes this service *after* it has invokes wm.tn.queuing:updateQueuedTask to dequeue a delivery task and has attempted to deliver the document associated with the delivery task. For an example of a scheduled delivery service, see the wm.tn.transport:batchFtp service. For a description about how to create a scheduled delivery service, including how to use the wm.tn.queuing:updateQueuedTask service, see *webMethods Trading Networks Administrator's Guide*.
- Your service should not update the status of the delivery task or its associated document. Trading Networks updates these statuses based on whether you indicate `success` or `fail` for *status*. Additionally, Trading Networks also maintains the retry count of the delivery task.

16 Route Folder

Overview

Use processing rule services (services in the wm.tn.route folder) to reload, lookup, and manually trigger document processing rules.

Summary of Elements in this Folder

The following table describes the elements that are available in this folder:

Element	Description
wm.tn.route:abort	Aborts processing of the current document.
wm.tn.route:add	Adds a new processing rule.
wm.tn.route:alert	Send an alert email message.
wm.tn.route:containsRuleName	Checks if the processing rule exists.
wm.tn.route:create	Creates a processing rule object.
wm.tn.route:delete	Deletes a processing rule.
wm.tn.route:disableDeliveryForPartner	Suspends or resumes delivery for a specified partner. When you suspend delivery for a partner, Trading Networks stops delivering documents to that partner.
wm.tn.route:getAllMatches	Retrieves all processing rules that match a document.
wm.tn.route:getAnyTask	Retrieves information about a task. If your Integration Server is in a clustered environment, the service can retrieve a task associated with any server in the cluster.
wm.tn.route:getLastChangeID	Retrieves the ID generated after making the last change to any rule.
wm.tn.route:getRule	Retrieves a single processing rule.
wm.tn.route:getTask	Retrieves a specified task that is associated with this server (the server on which the wm.tn.route:getTask service is being executed).
wm.tn.route:list	Retrieves the list of processing rules.
wm.tn.route:load	Reloads the processing rules from the database.
wm.tn.route:mergeFlags	Merges flags between document types to processing rules.
wm.tn.route:preroute	Calls wm.tn.doc:verify, wm.tn.doc:persist, and wm.tn.doc:validate services.

Element	Description
wm.tn.route:route	Processes the specified document using the specified processing rule.
wm.tn.route:routeBizdoc	Submits for processing a document that has already been recognized by Trading Networks; that is, submits a bizdoc (wm.tn.rec:BizDocEnvelope).
wm.tn.route:update	Updates the existing processing rule.

wm.tn.route:abort

Aborts processing of the current document.

Input Parameters

message	**String** A message to log with the document that describes the reason for aborting the processing of the document.
saveDocument	**String** (optional) Whether to save the document for which processing is being aborted to the database. Valid values are: ■ `true` - To save the document. ■ `false` - Default. To *not* save the document.
procstat	**String** (optional) The processing status the service assigns to the aborted document (for example, "ABORTED"). There is no default. If you do not specify *procstat*, the processing status remains unchanged.
userstat	**String** (optional) The user status the service assigns to the aborted document (for example, "ABORTED"). There is no default. If you do not specify *userstat*, the user status remains unchanged.

Output Parameters

None.

Usage Notes

- Invoke this service when you do not want Trading Networks to perform the remainder of processing actions (for example Deliver document or Respond with) for a document.
- It is recommended that you specify `true` for *saveDocument*. This makes it easier to debug the conditions leading up to the abort if a trail of log messages is available. If Trading Networks has already saved the document before this service is invoked, specify `false` for *saveDocument* has no effect.

wm.tn.route:add

Adds a new processing rule.

Input Parameters

rule	**Object** The processing rule that you want to add. The processing rule must be an instance of com.wm.app.tn.route.RoutingRule.
lastChangeID	**String** The ID generated after making the last change to any rule.

Output Parameters

lastChangeID	**String** The ID generated after adding the new rule.

wm.tn.route:alert

Send an alert email message.

Input Parameters

bizdoc	**Object** (optional) The document this email message pertains to (if any). The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope
InternalPartnerId	**String** Identifies to whom TN is to send the email message. This is either the internal identifier for a partner in your network or the value `B2B` to indicate the email message is being sent to the webMethods administrator.
partnerContact	**String** (optional) The contact at the partner's corporation who is to receive the email message (for example, "Administrative"). If the recipient of this email message is the webMethods administrator, leave this variable empty.
subject	**String** The subject of the email message.
body	**String** The body of the email message.

Output Parameters

sent	**String** Identifies the status of the delivered email message. Valid values are: ■ `true` The email message was sent successfully. ■ `false` The email message was *not* sent successfully.

sendError	**Document** A description of the error that prevented the email message from being sent (see wm.tn.rec:ActivityLogEntry). If you are invoking this service from a Java program, this is an instance of com.wm.app.tn.error.ActivityLogEntry.

Usage Notes

The Integration Server must have an SMTP server defined to delivery email messages.

wm.tn.route:containsRuleName

Checks if the processing rule exists.

Input Parameters

ruleName	**String** Processing rule to be checked.

Output Parameters

result	**String** Whether the processing rule exists. Valid values are: ■ `true` - The processing rule exists. ■ `false` - The processing rule does not exist.

wm.tn.route:create

Creates a processing rule object.

Input Parameters

name	**String** Name of the processing rule.
description	**String** (optional) Description of the processing rule.
disabled	**String** (optional) Status of the processing rule. Valid values are: ■ `true` - Processing rule is disabled. ■ `false` - Processing rule is not disabled.
senderId	**String** The internal partner ID of the sender of the document. Valid values are `Any`, `Unknown`, and `My Enterprise`.
receiverId	**String** The internal partner ID of the receiver of the document. Valid values are `Any`, `Unknown`, and `My Enterprise`.

docTypeId **String** Document type identifier for the resulting document. To determine the document type identifier of a document type, invoke the wm.tn.doctype:list service. The wm.tn.doctype:list service lists the names and IDs of all your document types.

userStatus **String** (optional) User status of the transaction.

hasErrors **String** (optional) Whether a transaction has errors. Valid values are:

- `yes` - Transaction has errors.
- `no` - Transaction does not have errors.
- `don't care` - Use the setting specified in the document type for the transaction.

verify **String** (optional) Whether to verify the signature of the transaction. Valid values are:

- `yes` - Transaction should be verified.
- `no` - Transaction should not be verified.
- `don't care` - Use the setting specified in the document type for the transaction.

validate **String** (optional) Whether to validate the structure of the transaction. Valid values are:

- `yes` - Transaction should be validated.
- `no` - Transaction should not be validated.
- `don't care` - Use the setting specified in the document type for the transaction.

persist **String** (optional) Whether to save the transaction to the database. Valid values are:

- `yes` - Transaction should be saved to the database.
- `no` - Transaction should not be saved to the database.
- `only if unique` - Transaction should be saved only if unique.
- `don't care` - Use the setting specified in the document type for the transaction.

persistOption? **String** (optional) The data that Trading Networks has to save. Valid values are:

- `content, attributes, and activity log` - Trading Networks saves all data associated with the transaction; that is, Trading Networks

saves the document content, the values extracted for the custom attributes, and the activity log entries.

- `content only` - Trading Networks saves only the document content. Trading Networks does not save the values of any extracted custom attributes or the related activity log entries.
- `attributes only` - Trading Networks saves only the values it extracts for the custom attributes. Trading Networks does not save the document content or the related activity log entries.
- `activity log only` - Trading Networks saves only the activity log entries. Trading Networks does not save the document content or the values of any extracted custom attributes.
- `content and attributes` - Trading Networks saves the document content and the values of all extracted custom attributes. Trading Networks does not save the related activity log entries.
- `content and activity log` - Trading Networks saves the document content and the activity log entries. Trading Networks does not save the values of extracted custom attributes.
- `attributes and activity log` - Trading Networks saves the values of all extracted custom attributes and the activity log entries. Trading Networks does not save the document content.
- `don't care` - Trading Networks refers to the settings specified in the processing rule to determine the data to save.
- `persist none` Trading Networks saves no data associated with the transaction.

unique

String (optional) Whether to check the uniqueness of a transaction.

Valid values are:

- `DocumentID only` - The document is checked for uniqueness. To determine if the document is unique, Trading Networks determines whether another document with the same document ID exists in the database. (The document ID is a user-defined, external identifier for the document.)
- `DocumentID and sender` - The document is checked for uniqueness. To determine if the document is unique, Trading Networks determines whether another document with the same document ID and sender exists in the database.
- `DocumentID, sender and receiver` - The document is checked for uniqueness. To determine if the document is unique, Trading Networks determines whether another document with the same document ID, sender, and receiver exists in the database.

- `DocumentID, sender and document type` - The document is checked for uniqueness. To determine if the document is unique, Trading Networks determines whether another document with the same document ID, sender, and TN document type exists in the database.
- `don't care` - Use the setting specified in the TN document type for the document being processed.

alertPartner **String** (optional) The partner to whom Trading Networks should send the email alert. Valid values are `B2B`, `SENDER`, and `RECEIVER`.

alertContactType **String** (optional) The contact type of the partner to whom Trading Networks sends the email alert. Valid values are `Administrative` and `Technical`. You can create additional ones with the wm.tn.dictionary:addContactType service.

alertSubject **String** (optional) Subject of the email alert.

alertMessage **String** (optional) Message sent in the email alert.

responseMessage **String** (optional) Response message received in response to the email alert.

responseType **String** (optional) Content type of the response message. Valid values are `text/plain` and `text/xml`.

newUserStatus **String** (optional) New user status of the transaction.

serviceName **String** (optional) Name of the service that Trading Networks will execute as a part of post processing action defined in a processing rule.

serviceInput **String** (optional) Inputs to the service.

serviceType **String** (optional) Type of the service. Valid values are `sync`, `async`, and `reliable`.

deliverUsing **String** (optional) The delivery protocol that the partner prefers you to use while sending documents. Valid values are:

- Preferred protocol
- Primary HTTP
- Primary HTTPS
- Primary Email
- Primary FTP
- Secondary HTTP
- Secondary HTTPS
- Secondary Email

- Secondary FTP
- Queue for Polling
- Queue for Delivery

deliveryQueue **String** (optional) Registered scheduled delivery queue. Valid value is `Receiver's queue`.

attribConditions **String** (optional) Attribute criteria to use to match processing rule.

attribute **String** Internal ID of the custom attribute.

operation **String** Operation to use in matching processing rules. Valid values are:

- IS NULL
- IS NOT NULL
- =
- <>
- >=
- <=
- >
- <
- CONTAINS
- BEFORE
- AFTER
- INCLUDES

value **String** (optional) Value to check while matching processing rules.

Output Parameters

rule **Object** Processing rule created.

wm.tn.route:delete

Deletes a processing rule.

Input Parameters

ruleIDs	**Object** The IDs of the processing rules that you want to delete.
lastChangeID	**String** The ID generated after making the last change to any rule.

Output Parameters

lastChangeID	**String** The ID generated after adding the new rule.

wm.tn.route:disableDeliveryForPartner

Suspends or resumes delivery for a specified partner. When you suspend delivery for a partner, Trading Networks does not deliver documents to that partner.

Input Parameters

profileID	**String** The internal ID of the profile (partner) to whom delivery of documents is to be suspended or resumed.
disabled	**String** Whether you want to suspend or resume delivery. The following values apply: ■ `true` Suspend delivery. Default. ■ `false` Resume delivery.

Output Parameters

errorMessage	**String** An error message that resulted from executing the service, if any.

wm.tn.route:getAllMatches

Returns all processing rules that match the current document available in the pipeline with *BizDocAttributes* map passed in the service input.

Input Parameters

attrs	**Object** A map of key and value pairs of *BizDocAttributes*.

Output Parameters

matches	**String** A list of processing rule internal ID matches that adheres to the specified criteria.

wm.tn.route:getAnyTask

Retrieves information about a task. If your Integration Server is in a clustered environment, the service can retrieve a task associated with any server in the cluster.

Input Parameters

taskId	**String** The internal identifier for the task that you want to retrieve.
includeBizDocErrors	**String** Prevents the Envelope (see "wm.tn.rec:BizDocEnvelope" on page 331) associated with the task from having the Errors ("wm.tn.rec:BizDocErrorSet" on page 334) field populated with Activity Log error messages. Valid values are: ■ `true` - Default. Envelope is loaded with errors from the Activity Log. ■ `false` - Envelope is not loaded with errors from the Activity Log.

Output Parameters

task	**Document** The task identified by taskId that this service retrieved. For the structure of task, see wm.tn.rec:Task.

Usage Notes

- If taskId is not valid, the service throws an exception.
- The service retrieves any task regardless of the server on which the task was started.
- The service might run slower than wm.tn.task:getTask because it might have to retrieve the information from the Trading Networks database.
- If you do not need errors in Envelope, set the *includeBizDocErrors* parameter to false to decrease the retrieval time and memory usage.

wm.tn.route:getLastChangeID

Retrieves the ID generated after making the last change to any rule.

Input Parameters

None.

Output Parameters

lastChangeID	**String** The ID generated after making the last change to any rule..

wm.tn.route:getRule

Retrieves a single processing rule.

Input Parameters

ruleID	**String** (optional) The internal unique identifier of the processing rule that you want to retrieve.
ruleName	**String** (optional) The name of the processing rule that you want to retrieve.

Output Parameters

rule	**Object** The processing rule that you want to retrieve. The processing rule must be an instance of com.wm.app.tn.route.RoutingRule..

Usage Notes

- You must supply either the ruleID or ruleName. If you supply both, Trading Networks uses ruleID.
- If you are invoking this service from a Java program, in addition to returning rule as an Integration Server document (IData object), the service returns rule as an instance of com.wm.app.tn.route.RoutingRule.

wm.tn.route:getTask

Retrieves a specified task that is associated with this server (the server on which the wm.tn.route:getTask service is being executed).

Input Parameters

taskID	**String** The internal unique identifier of the task that you want to retrieve.

includeBizDocErrors	**String** Prevents the Envelope (see "wm.tn.rec:BizDocEnvelope" on page 331) associated with the task from having the Errors ("wm.tn.rec:BizDocErrorSet" on page 334) field populated with Activity Log error messages. Valid values are: ■ `true` - Default. Envelope is loaded with errors from the Activity Log. ■ `false` - Envelope is not loaded with errors from the Activity Log.

Output Parameters

task	**Document** The task identified by taskID that the service retrieves. For the structure of task, see wm.tn.rec:Task.

Usage Notes

- If taskID is not valid, the service throws an exception.
- Each task is associated with a single server, so taskID must be associated with the same server on which the task was started.
- To get a task that was started on another server, see the wm.tn.route:getAnyTask service.
- If you do not need errors in Envelope, set the *includeBizDocErrors* parameter to false to decrease the retrieval time and memory usage.

wm.tn.route:list

Retrieves the list of processing rules.

Input Parameters

refresh	**String** (optional) Whether you want to refresh the document attribute cache, before returning the list of document attributes. Valid values are: ■ `true` - Refreshes the cache. ■ `false` - Does not refresh the cache. Default.

Output Parameters

ruleCount	**String** The number of processing rules that are defined..
rules	**String** List of processing rules.
lastChangeID	**String** The ID generated after making the last change to any rule.

wm.tn.route:load

Reloads the processing rules from the database.

Input Parameters

None

Output Parameters

ruleCount	**String** The number of processing rules that are defined.

wm.tn.route:mergeFlags

Merges flags between document types to processing rules.

Input Parameters

bizdoc	**Document** The source of flags are retrieved from the document type of the document. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
rule	**Object** The destination of flags are merged to the processing rule. The processing rule must be an instance of com.wm.app.tn.route.RoutingRule.

Output Parameters

merge	**Document** Retrieves the merged flags. For the structure of merge flags, see wm.tn.rec:PreProcessingFlags.

wm.tn.route:preroute

Calls wm.tn.doc:verify, wm.tn.doc:persist, and wm.tn.doc:validate services.

Input Parameters

bizdoc	**Object** The recognized document that you want Trading Networks to pre-route. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
flags	**Object** Flags for the document that you want to pre-route.

wm.tn.route:route

Processes the specified document using the specified processing rule.

Input Parameters

bizdoc	**Object** The document to process. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
rule	**Object** The processing rule to use to process the specified document. The processing rule must be an instance of com.wm.app.tn.route.RoutingRule.

Output Parameters

The output that the service returns depends on the processing actions (specified in *rule*) that the service executed.

If *rule* specifies…	The wm.tn.route:route service returns:
`Execute a Service` action (but not the `Respond with` action)	The output from the executed service
`Respond with` action	*$-responseTime* and *$responseBytes* if the service was invoked by a client (for example, a Java client or C/C++ client). If the service was invoked by a browser client, the wm.tn.route:route service returns nothing.
Other	Nothing

wm.tn.route:routeBizdoc

Submits for processing a document that has already been recognized by Trading Networks; that is, submits a bizdoc (wm.tn.rec:BizDocEnvelope).

This service does *not* check the identity of the sender against the currently logged in user. Only invoke this service from within processing rules or services; do *not* expose directly to trading partners. Trading partners should use wm.tn:submit.

Input Parameters

bizdoc	**Object** The recognized document that you want Trading Networks to process. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
TN_parms	**Document** (optional) If either of the nested elements *processingRuleID* or *processingRuleName* are present, Trading Networks bypasses the processing

rule matching process and instead uses the specified rule to process the bizdoc. If both *processingRuleID* and *processingRuleName* are present, Trading Networks uses the *processingRuleID* to determine which rule to use. If neither are present, Trading Networks executes the rule matching process as described in the chapter about processing rules in *webMethods Trading Networks Administrator's Guide*.

The following are the pipeline variables for this document.

- *processingRuleID*- **String** (optional) The internal identifier of the processing rule that should be used to process this *bizdoc*.
- *processingRuleName*- **String** (optional) The name of the processing rule that should be used to process this *bizdoc*.

Output Parameters

bizdoc	**Document** The document. For the structure of *bizdoc*, see wm.tn.rec:BizDocEnvelope.
sender	**Document** The profile summary for the sender of the document. For the structure of *sender*, see wm.tn.rec:ProfileSummary.
receiver	**Document** The profile summary for the receiver of the document. For the structure of *receiver*, see wm.tn.rec:ProfileSummary.

Usage Notes

- This service is protected by the TNAdministrators ACL.
- To submit a bizdoc externally, use the wm.tn:submit service.
- If you are invoking this service from a Java program, in addition to returning *bizdoc*, *sender*, and *receiver* as IS documents (IData objects), the service returns *bizdoc* as an instance of com.wm.app.tn.doc.BizDocEnvelope and the returned *sender* and *receiver* as instances of com.wm.app.tn.profile.ProfileSummary.
- This service returns after Trading Networks completes processing for the document. That is, after Trading Networks has executed the pre-processing and processing actions for the document. If the processing actions instructed Trading Networks to execute a service asynchronously, the asynchronously invoked service may not be complete.
- If you are submitting documents to Trading Networks from an internal application, that application might know which processing rule should be used to process the document. In this case, you might improve the performance of your Trading Networks application (reduce latency and/or increase document throughput) by explicitly stating which processing rule should be used to process the document. To do this, specify *processingRuleID* or *processingRuleName* in the *TN_parms* document in the pipeline when you submit the document

to Trading Networks from the internal application. See the description of the input signature for details.

wm.tn.route:update

Updates the existing processing rule.

Input Parameters

rule	**Object** The processing rule that you want to update. The processing rule must be an instance of com.wm.app.tn.route.RoutingRule.

17 Security Folder

Overview

Use security services (services in the wm.tn.security folder) to:

- Retrieve certificates and private keys for signing, encryption, verification, and decryption purposes.
- Add, update, and delete certificates or private keys for signing, encryption, verification, and decryption purposes.

You can set (and subsequently retrieve) the certificate information to use between any two partners in the network.

You can specify up to two certificate sets for any owner, partner, and usage combination. The certificate set that you add first is the *primary* certificate set. Trading Networks uses the primary certificate set until it expires. When a primary sign/verify, encrypt/decrypt, or SSL certificate set expires, Trading Networks automatically sets the other certificate pair as the primary certificate.

Summary of Elements in this Folder

The following table describes the elements that are available in this folder:

Element	Description
wm.tn.security:addCertificateData	Adds a signing, decryption, or SSL client certificate to the Trading Networks database.
wm.tn.security:deleteCertificateData	Deletes a signing, decryption, or SSL client certificate from the Trading Networks database.
wm.tn.security:getAllCertificateData	Retrieves information about active and inactive certificates for a given combination of owner, partner, and usage.
wm.tn.security:getCertificateData	Retrieves certificate data from the Trading Networks database.
wm.tn.security:getDecryptionKeyAndCert	Retrieves the decryption private key and certificates of a document receiver.
wm.tn.security:getEncryptionChain	Retrieves the encryption and signing certificates of a document receiver.
wm.tn.security:getSigningKeyAndChain	Retrieves the signing private key and certificates of the document receiver.
wm.tn.security:getSSLKeyAndChain	Retrieves a client's SSL private key and certificates.
wm.tn.security:getVerifyingChain	Retrieves the verifying certificate and certificate chain of a document sender.

Element	Description
wm.tn.security:queryExpiredCertificates	Retrieves certificates that are expiring soon and optionally certificates that have already expired.
wm.tn.security:setPrimaryCertificate	Sets a certificate as the primary certificate.
wm.tn.security:setSSLKeyAndChain	Retrieves the SSL client certificate info from the database, and associates the private key and certificate chain with the subsequent set of invoked services.
wm.tn.security:updateCertificate	Replaces an existing certificate set with a new certificate set.
wm.tn.security:updateCertificateData	Updates certificate data based on certificate ID or owner, partner, and usage combination.

wm.tn.security:addCertificateData

Adds a signing, decryption, or SSL client certificate to the Trading Networks database.

Input Parameters

ownerID **String** The internal partner ID of the owner of the certificate.

partnerID **String** (optional) The internal partner ID of the certificate owner's partner.

See the Usage Notes at the end of this service's description for information about how the certificate data is used if *partnerID* is not specified.

usage **String** Specify one of the following values:

- `sign` - The certificate is used as a signing certificate for the owner to send digitally signed documents to the partner.
- `decrypt` - The certificate is used as an encrypt certificate for the owner to send encrypted documents to the partner.
- `ssl` - The certificate is used as a client SSL certificate for the owner to establish an SSL connection to the partner's secure server.

chainBytes **Object** (optional) An array of byte arrays. Each byte array should represent a java.security.cert.X509Certificate. The certificates should be in node-to-root order. The first certificate in the array should be the signing, decryption, or SSL client certificate. Each subsequent certificate should be the certificate that was used to sign the previous certificate in the array.

keyBytes **Object** (optional) A byte array that represents the private key that is used to generate the certificate.

Note:
If you are adding certificate data for your Enterprise profile, specify the private key using *keyAliasName*, not *keyBytes*.

keyStoreAliasName **String** (optional) Alias for the keystore file associated with the certificate.

Note:
Keystores apply only to Enterprise profiles. If you are adding certificate data for your Enterprise profile, supply a value for this parameter. If you are adding certificate data for a partner profile, leave this parameter blank.

keyAliasName **String** (optional) Configured private key alias in the specified keystore.

Note:
Key aliases apply only to Enterprise profiles. If you are adding certificate data for a partner profile, specify the private key using *keyBytes*, not *keyAliasName*.

Output Parameters

certID **String** The internal ID that uniquely identifies the certificate data that is added.

addCount **String** Whether the certificate data is successfully inserted into the Trading Networks database. The value `1` indicates success. A null value indicates failure.

Usage Notes

- This service is only used for adding new certificate data. If certificate data already exists for the specified *ownerID*/*partnerID* usage, use wm.tn.security:updateCertificateData.
- If both *ownerID* and *partnerID* are specified, the certificate data is used for the purpose you specify in *usage*. If *partnerID* is not specified (or no specific alternative certificate data is defined by the owner and the specified partner for the purpose you specify in *usage*), the certificate data is used as a default certificate set for the owner and all of the owner's partners.

wm.tn.security:deleteCertificateData

Deletes a signing, decryption, or SSL client certificate from the Trading Networks database.

Input Parameters

certID **String** (optional) The internal ID that uniquely identifies the certificate data to be deleted.

See the Usage Notes at the end of this service's description for information about the relationship between the *certID*, *ownerID*, *partnerID*, and *usage* parameters.

ownerID **String** (optional) The internal partner ID of the owner of the certificates.

See the Usage Notes at the end of this service's description for information about the relationship between the *certID*, *ownerID*, *partnerID*, and *usage* parameters.

partnerID **String** (optional) The internal partner ID of the certificate owner's partner.

If *partnerID* is not specified, the default certificate set is deleted.

See the Usage Notes at the end of this service's description for information about the relationship between the *certID*, *ownerID*, *partnerID*, and *usage* parameters.

usage **String** Specify how the certificate is used:

- `sign` - A signing certificate for the owner to send digitally signed documents to the partner.
- `decrypt` - An encrypt certificate for the owner to send encrypted documents to the partner.
- `ssl` - A client SSL certificate for the owner to establish an SSL connection to the partner's secure server.

Output Parameters

deleteCount **String** Whether the certificate data is successfully deleted from the Trading Networks database. The value `1` indicates success. A null value indicates failure.

Usage Notes

- Be aware that the certificate data you delete might be a default certificate set, depending on how the certificate data was defined by wm.tn.security:addCertificateData.
- If *certID* is specified, *ownerID*, *partnerID*, and *usage* are ignored. If *certID* is not specified, you must specify *ownerID*, *partnerID*, and *usage*.

wm.tn.security:getAllCertificateData

Retrieves information about active and inactive certificates for a given combination of owner, partner, and usage.

Input Parameters

ownerID **String** The internal partner ID of the owner of the certificates.

partnerID **String** (optional) The internal partner ID of the certificate owner's partner.

See the Usage Notes at the end of this service's description for information about how the certificate data is used if *partnerID* is not specified.

usage **String** Specify one of the following values:

- `sign` - The private key is used as a signing key for the owner to send digitally signed documents to the partner. The public key is used by the partner to verify the signed document.
- `decrypt` - The private key is used to decrypt the encrypted document sent to the owner from the partner. Public certificates are used to encrypt the document by the partner.
- `ssl` - The certificate is used as a client SSL certificate for the owner to establish an SSL connection to the partner's secure server.

Output Parameters

certIData **Document List** An array of IData objects each with the following fields:

Key	Description
certID	**String** The internal ID that uniquely identifies the certificate data that is retrieved. If the input parameter *certID* is specified, the same *certID* appears in the output pipeline.
ownerID	**String** The internal partner ID of the owner of the certificate.
partnerID	**String** The internal partner ID of the certificate owner's partner.
usage	**String** Indicates how the certificate is used: - `sign` - The private key is used as a signing key for the owner to send digitally signed documents to the partner. The public key is used by the partner to verify the signed document. - `decrypt` - The private key is used to decrypt the encrypted document sent to the owner from the partner. Public certificates are used by the partner to encrypt the document.

- `ssl` - The certificate is used as a client SSL certificate for the owner to establish an SSL connection to the partner's secure server.

chainBytes

Object An array of byte arrays. Each byte array represents a java.security.cert.X509Certificate. The certificates are in node-to-root order. The first certificate in the array is the sign/decrypt/SSL client certificate. Each subsequent certificate is the certificate that was used to sign the previous certificate in the array.

keyBytes

Object A byte array that represents the private key that is used to generate the certificate.

expirationDate

String The expiration date of the certificate.

priority

String Identifies whether the certificate is the primary or secondary certificate, as follows:

- `0` - The certificate is the primary (active) certificate.
- 1 - The certificate is the secondary (inactive) certificate.

keyStoreAlias

String Alias for the keystore file associated with the certificate.

Note:
Keystores apply only to Enterprise profiles. If you are adding certificate data for your Enterprise profile, supply a value for this parameter. If you are adding certificate data for a partner profile, leave this parameter blank.

keyAlias

String Configured private key alias in the specified keystore.

Note:
Key aliases apply only to Enterprise profiles. If you are adding certificate data for a partner profile, specify the private key using *keyBytes*, not *keyAlias*.

Usage Notes

- If both *ownerID* and *partnerID* are specified, the certificate data is used for the purpose you specify in *usage*. If *partnerID* is not specified (or no specific alternative certificate data is defined by the owner and the specified partner for the purpose you specify in *usage*), the certificate data is used as a default certificate set for the owner and all of the owner's partners.

wm.tn.security:getCertificateData

Retrieves certificate data from the Trading Networks database.

Input Parameters

certID **String** (optional) The internal ID that uniquely identifies the certificate data that is to be retrieved.

See the Usage Notes at the end of this service's description for information about the relationship between the *certID*, *ownerID*, *partnerID*, and *usage* parameters.

ownerID **String** (optional) The internal partner ID of the owner of the certificates.

See the Usage Notes at the end of this service's description for information about the relationship between the *certID*, *ownerID*, *partnerID*, and *usage* parameters.

partnerID **String** (optional) The internal partner ID of the certificate owner's partner.

See the Usage Notes at the end of this service's description for information about the relationship between the *certID*, *ownerID*, *partnerID*, and *usage* parameters.

usage **String** Specify how the certificate is used:

- `sign` - A signing certificate for the owner to send digitally signed documents to the partner.
- `decrypt` - An encrypt certificate for the owner to send encrypted documents to the partner.
- `ssl` - A client SSL certificate for the owner to establish an SSL connection to the partner's secure server.

See the Usage Notes at the end of this service's description for information about the relationship between the *certID*, *ownerID*, *partnerID*, and *usage* parameters.

Output Parameters

certID **String** The internal ID that uniquely identifies the certificate data that is retrieved. If the input parameter *certID* is specified, the same *certID* appears in the output pipeline.

ownerID **String** The internal partner ID of the owner of the certificates.

partnerID **String** The internal partner ID of the certificate owner's partner.

usage	**String** Indicates how the certificate is used: - `sign` - A signing certificate for the owner to send digitally signed documents to the partner. - `decrypt` - An encrypt certificate for the owner to send encrypted documents to the partner. - `ssl` - A client SSL certificate for the owner to establish an SSL connection to the partner's secure server.
chainBytes	**Object** An array of byte arrays. Each byte array represents a java.security.cert.X509Certificate. The certificates are in node-to-root order. The first certificate in the array is the sign/decrypt/SSL client certificate. Each subsequent certificate is the certificate that was used to sign the previous certificate in the array.
keyBytes	**Object** A byte array that represents the private key that is used to generate the certificate.
expirationDate	**Object** The expiration date of the certificate.

Usage Notes

- If both *ownerID* and *partnerID* are specified, the certificate data is used for the purpose you specify in *usage*. If *partnerID* is *not* specified (or no specific alternative certificate data is defined by the owner and the specified partner for the purpose you specify in *usage*), the certificate data is used as a default certificate set for the owner and *all* of the owner's partners.
- If *certID* is specified, *ownerID*, *partnerID*, and *usage* are ignored. If *certID* is not specified, you must specify *ownerID*, *partnerID*, and *usage*.
- If *certID* is not specified and a secondary certificate has been provided, this service switches the certificates when the primary certificate expires.

wm.tn.security:getDecryptionKeyAndCert

Retrieves the decryption private key and certificates of a document receiver.

Input Parameters

senderID	**String** (optional) The internal partner ID of the document sender. If *senderID* is not specified (or if no specific alternative decryption certificate data is defined between the sender and the receiver), the service will retrieve the receiver's default decryption private key and certificates.
receiverID	**String** The internal partner ID of the document receiver (the owner of the decryption certificates).

Output Parameters

senderID	**String** The internal partner ID of the document sender. If *senderID* is null but the output parameter *recipientCert* is not null, the service retrieves the receiver's default decryption certificate.
receiverID	**String** The internal partner ID of the document receiver (the owner of the decryption certificates).
privKey	**Object** A byte array that represents the private key that is used to generate the decryption certificate.
recipientCert	**Object** A byte array that represents the decryption certificate (a java.security.cert.X509Certificate).
recipientCertChain	**Object** An array of byte arrays. Each byte array represents a java.security.cert.X509Certificate. The certificates are in node-to-root order. The first certificate in the array is a decryption certificate. Each subsequent certificate is the certificate that was used to sign the previous certificate in the array.
isDefault	**String** Whether the decryption certificate is the receiver's default decryption certificate. Valid values are: ■ `true` - The decryption certificate is the recipient's default decryption certificate that the document receiver uses to decrypt all incoming documents if no alternative decryption certificate is defined between the receiver and the sender. ■ `false` - The decryption certificate is not the recipient's default decryption certificate.

wm.tn.security:getEncryptionChain

Retrieves the encryption and signing certificates of a document receiver.

Input Parameters

senderID	**String** The internal partner ID of the document sender.
receiverID	**String** The internal partner ID of the document receiver (the owner of the encryption certificates).

Output Parameters

senderID	**String** The internal partner ID of the document sender.

receiverID	**String** The internal partner ID of the document receiver (the owner of the encryption certificates).
recipientCert	**Object** An array of byte arrays. Each byte array represents a java.security.cert.X509Certificate. The certificates are in node-to-root order. The first certificate in the array is the encryption certificate. Each subsequent certificate is the certificate that was used to sign the previous certificate in the array.
isDefault	**String** Whether the encryption certificate is the receiver's default encryption certificate. Valid values are: ■ `true` - The encryption certificate is the recipient's default certificate. All senders use this to encrypt outgoing documents to this receiver if no alternative encryption certificate is defined between the receiver and sender). ■ `false` - The encryption certificate is not the recipient's default certificate.

wm.tn.security:getSigningKeyAndChain

Retrieves the signing private key and certificates of the document receiver.

Input Parameters

senderID	**String** The internal partner ID of the document sender.
receiverID	**String** (optional) The internal partner ID of the document receiver. If *receiverID* is not specified (or if no specific alternative signing certificate data is defined between the sender and the receiver), the service retrieves the sender's default signing private key and certificates.

Output Parameters

senderID	**String** The internal partner ID of the document sender.
receiverID	**String** The internal partner ID of the document receiver. If *receiverID* is null, the key and certificates are the sender's default key and certificates.
key	**Object** A byte array that represents the private key used to generate the signing certificate.
certChain	**Object** An array of byte arrays. Each byte array represents a java.security.cert.X509Certificate. The certificates are in node-to-root order. The first certificate in the array is the sign/decrypt/SSL client certificate. Each subsequent certificate is the certificate that was used to sign the previous certificate in the array.

cert	**Object** A byte array that represents the signing certificate (a java.security.cert.X509Certificate).
isDefault	**String** Whether the signing certificates are the sender's default signing certificates. Valid values are: ■ `true` - The certificate is the sender's default certificate. The document receiver uses this to decrypt all outgoing documents if no alternative signing certificate is defined between the receiver and the sender. ■ `false` - The certificate is not the sender's default certificate.

wm.tn.security:getSSLKeyAndChain

Retrieves a client's SSL private key and certificates.

Input Parameters

clientID	**String** The internal ID of the partner that acts as the client in the SSL connection.
serverID	**String** (optional) The internal ID of the partner that acts as the remote server in the SSL connection.

Output Parameters

clientID	**String** The internal ID of the partner that acts as the client in the SSL connection.
serverID	**String** The internal ID of the partner that acts as the remote server in the SSL connection. If *serverID* is null, that indicates that the key and certificate that were retrieved are the default SSL client key and certificate.
key	**Object** A byte array that represents the private key that is used to generate the SSL client certificate.
certChain	**Object** An array of byte arrays. Each byte array represents a java.security.cert.X509Certificate. The certificates are in node-to-root order. The first certificate in the array is the client SSL certificate. Each subsequent certificate is the certificate that was used to sign the previous certificate in the array.
whichSSLCert	**String** Indicates one of the following values for the SSL certificate: ■ `TN_alternative` - The SSL client certificate, saved in the Trading Networks database, is specific to the client and remote server.

- `TN_default` - The SSL client certificate, saved in the Trading Networks database, is the default client SSL certificate for SSL connections to all remote secure servers.
- `IS_default` - The certificate saved in Integration Server as "Outbound SSL certificates."

wm.tn.security:getVerifyingChain

Retrieves the verifying certificate and certificate chain of a document sender.

Input Parameters

senderID	**String** The internal partner ID of the document sender (the owner of the verifying certificate).
receiverID	**String** (optional) The internal partner ID of the document sender. If *receiverID* is not specified (or if no specific alternative verifying certificate is defined between the sender and the receiver), the service will retrieve the sender's default verifying certificates.

Output Parameters

senderID	**String** The internal partner ID of the document sender.
receiverID	**String** The internal partner ID of the document sender. If *receiverID* is null, that indicates that the certificates that were retrieved are the sender's default verifying certificates.
certChain	**Object** An array of byte arrays. Each byte array represents a java.security.cert.X509Certificate. The certificates are in node-to-root order. The first certificate in the array is the verifying certificate. Each subsequent certificate is the certificate that was used to sign the previous certificate in the array.
cert	**Object** A byte array that represents the signing certificate (a java.security.cert.X509Certificate).
isDefault	**String** Whether the verifying certificate is the sender's default verifying certificate. Valid values for the verifying certificate: - `true` - This is the sender's default signing certificate that the sender uses to sign all outgoing documents if no alternative signing certificate exists between the sender and receiver. - `false` - This is not the sender's default signing certificate.

wm.tn.security:queryExpiredCertificates

Retrieves certificates that are about to expire and optionally certificates that have already expired.

Input Parameters

numOfDaysToExpire **String** The number of days (including today) until the certificates expire.

inclExpiredCerts **String** (optional) Whether to return information about certificates that have already expired. Specify one of the following:

- `true` - Include expired certificates.
- `false` - Do not include expired certificates. This is the default.

Output Parameters

output **Document** The information about the certificates that are expiring. The document contains the following variables:

- *certinfo* **Document List** Returns the following fields for each certificate in the output:

Key	Description
certId	**String** The internal certificate ID.
fromPartnerID	**String** The internal partner ID for the partner defined as the sending partner for the certificate.
fromPartnerCorporationName	**String** The corporation name of the sending partner for the certificate.
fromPartnerOrgUnitName	**String** The organization unit of the sending partner for the certificate. Null if not defined.
toPartnerID	**String** The internal partner ID of the receiving partner for the certificate.
toPartnerCorporationName	**String** The corporation name of the receiving partner for the certificate.

toPartnerOrgUnitName	**String** The organization unit of the receiving partner for the certificate. Null if not defined.
usage	**String** How the certificate is used. Valid values are sign (for Sign/Verify), decrypt (for Encrypt/Decrypt), and ssl (for SSL connections).
subject	**String List** The subject from the certificate.
expirationDate	**String** The expiration date of the certificate, specified in Coordinated Universal Time (UTC) format (for example, `Monday, 2007-07-09T03:25UTC`).

- *errorMessages* **String List** Error messages for errors that the service encountered during execution, if any.

wm.tn.security:setPrimaryCertificate

Sets a certificate as the primary certificate.

Input Parameters

ownerID	**String** The owner ID of the owner of the certificate to be made primary.
partnerID	**String** (optional) The partner ID of the owner's partner for the certificate to be made primary.
usage	**String** Usage of the certificate to be made primary. Specify one of the following values: - `sign` - The private key is used as a signing key for the owner to send digitally signed documents to the partner. The public key is used by the partner to verify the signed document. - `decrypt` - The private key is used to decrypt the encrypted document sent to the owner from the partner. Public certificates are used by the partner to encrypt the document. - `ssl` - The certificate is used as a client SSL certificate for the owner to establish an SSL connection to the partner's secure server.
certID	**String** Certificate ID of the certificate which is to be made primary.

Output Parameters

isUpdated	**String** Indicates if the certificate with the given ID was made the primary certificate, as follows: ■ **true** - The service was successful. ■ **false** - The service was not successful.

Usage Notes

- Before using this service to set a service as the primary certificate, use the wm.tn.security:addCertificateData service to get the certificate ID of a valid certificate for the owner, partner, and usage.
- This service is useful in cases where both of the decryption certificates are valid on the receiver side and the document that is received was encrypted with the secondary certificate. After ascertaining the correct decryption certificate, the certificate is set as the primary.

wm.tn.security:setSSLKeyAndChain

Retrieves the SSL client certificate information from the database and associates the private key and certificate chain with the subsequent set of invoked services.

For more information, see pub.security:setKeyAndChain and pub.security:clearKeyAndChain services in *webMethods Integration Server Built-In Services Reference*.

Input Parameters

clientID	**String** The internal ID of the partner that acts as the client in the SSL connection.
serverID	**String** The internal ID of the partner that acts as the remote server in the SSL connection.

Output Parameters

None.

Usage Notes

- Use this service to associate a key and certificate chain that is different from the Integration Server's default settings. For more information, see the Usage Notes for the pub.security:clearKeyAndChain service in *webMethods Integration Server Built-In Services Reference*.
- The service first looks for SSL client certificate for the specified client and server specified in *TN_alternative* (a variable in the wm.tn.security:getSSLKeyAndChain service). If one is not defined,

the service then looks for a *TN_alternative* SSL client certificate. If neither certificate is defined, the service does nothing; the outbound SSL certificates defined in the Integration Server is used. See the Usage Notes for wm.tn.security:getSSLKeyAndChain.

wm.tn.security:updateCertificate

Replaces an existing certificate set with a new certificate set. For example, you might use this service to update an owner's certificate set when an existing certificate set is about to expire.

Input Parameters

certOwnerId **String** The internal partner ID of the certificate owner. For signing and verifying certificates, specify the internal partner ID of the sender. For decryption and encryption certificates, specify the internal partner ID of the receiver.

oldCertInfo **Document** The existing certificate information to replace. The document contains the following variables:

Key	Description
privateKey	**Object** (optional) A byte array that represents the private key to be replaced.
cert	**Object** (optional) A byte array that represents the certificate to be replaced.
CACerts	**Object** (optional) An array of byte arrays, each of which represents a CA Certificate to be replaced, in node-to-root order.
chainBytes	**Object** (optional) An array of byte arrays. The first byte array in the array represents *cert*. The remaining bytes represent *CACerts* (in the same node-to-root order). **Note:** If *chainBytes* is specified, *cert* and *CACerts* values are ignored. If *chainBytes* is null, both *cert* and *CACerts* values are required.

newCertInfo **Document** The new certificate information for *certOwnerId.*The document contains the following variables:

Key	Description
privateKey	**Object** A byte array that represents the new private key.

cert	**Object** A byte array that represents the new certificate.
CACerts	**Object** An array of byte arrays, each of which represents the CA Certificate, in node-to-root order.
chainBytes	**Object** (optional) An array of byte arrays. The first byte array in the array represents *cert*. The remaining bytes represent *CACerts* (in the same node-to-root order).

Note:
If *chainBytes* is specified, *cert* and *CACerts* values are ignored. If *chainBytes* is null, both *cert* and *CACerts* values are required.

Output Parameters

errors	**String List** (optional) Any errors that occurred while updating the certificate information. Each string in the string list is a separate error that was encountered. The *errors* variable is not in the pipeline if no errors were found.

Usage Notes

Before you invoke this service, you should back up your database. This service introduces permanent changes to the database as it replaces the existing certificate information with the new certificate information.

wm.tn.security:updateCertificateData

Updates certificate data based on certificate ID or owner, partner, and usage combination.

Input Parameters

certID	**String** (optional) The internal ID that uniquely identifies the certificate data to be updated. See the Usage Notes at the end of this service's description for information about the relationship between the *certID*, *ownerID*, *partnerID*, and *usage* parameters.
ownerID	**String** (optional) The internal partner ID of the owner of the certificates.

See the Usage Notes at the end of this service's description for information about the relationship between the *certID*, *ownerID*, *partnerID*, and *usage* parameters.

partnerID

String (optional) The internal partner ID of the partner with whom the certificate owner does business using the certificates for the specific usage. See the Usage Notes at the end of this service's description for information about how the updated certificate data is used if *partnerID* is not specified.

See the Usage Notes at the end of this service's description for information about the relationship between the *certID*, *ownerID*, *partnerID*, and *usage* parameters.

usage

String Specify one of the following values:

- `sign` - The certificate is used as a signing certificate for the owner to send digitally signed documents to the partner.
- `decrypt` - The certificate is used as an encrypt certificate for the owner to send encrypted documents to the partner.
- `ssl` - The certificate is used as a client SSL certificate for the owner to establish an SSL connection to the partner's secure server.

See the Usage Notes at the end of this service's description for information about the relationship between the *certID*, *ownerID*, *partnerID*, and *usage* parameters.

chainBytes

Object An array of byte arrays. Each byte array should represent a java.security.cert.X509Certificate. The certificate should be in node-to-root order. The first certificate in the array should be the sign/decrypt/SSL client certificate. Each subsequent certificate should be the certificate that was used to sign the previous certificate in the array.

keyBytes

Object (optional) A byte array that represents the private key that is used to generate the certificate.

Note:
If you are updating certificate data for your Enterprise profile, specify the private key using *keyAliasName*, not *keyBytes*.

keyStoreAliasName

String (optional) Alias for the keystore containing the certificate to be updated.

Note:
Keystores apply only to Enterprise profiles. If you are updating certificate data for your Enterprise profile, supply a value for this parameter. If you are updating certificate data for a partner profile, leave this parameter blank.

keyAliasName	**String** (optional) Alias for the private key used to access this certificate in the specified keystore. **Note:** Key aliases apply only to Enterprise profiles. If you are updating certificate data for a partner profile, specify the private key using *keyBytes*, not *keyAliasName*.

Output Parameters

certID	**String** The internal ID that uniquely identifies the certificate data that is updated.
updateCount	**String** Whether to show how many rows were successfully updated in the Trading Networks database. The value `1` indicates to show the number of rows.

Usage Notes

- This service is only used for updating certificate data. If no certificate data exists for the specified *ownerID*/*partnerID*/*usage*, use wm.tn.security:addCertificateData.
- If *certID* is specified, *ownerID*, *partnerID*, and *usage* are ignored. If *certID* is not specified, you must specify *ownerID*, *partnerID*, and *usage*.
- If both *ownerID* and *partnerID* are specified, the updated certificate data is used for the purpose you specify in *usage*. If *partnerID* is *not* specified (or no specific alternative certificate data is defined by the owner and the specified partner for the purpose you specify in *usage*), the updated certificate data is used as a default certificate set for the owner and *all* of the owner's partners.

18 Task Folder

Overview

Use the task services (services in the wm.tn.task folder) to manage delivery tasks and service execution tasks.

Summary of Elements in this Folder

The following table describes the elements that are available in this folder:

Element	Description
wm.tn.task:changeServerForTask	Moves a task from one server to another when using Trading Networks in a clustered server environment.
wm.tn.task:getAnyTask	Retrieves information about a task. If your Integration Server is in a clustered environment, this service can retrieve a task associated with any server in the cluster.
wm.tn.task:getTask	Retrieves a specified task that is associated with this server (the server on which the wm.tn.task:getTask service is being executed).
wm.tn.task:getTaskOutput	Retrieves the output of a specified task.
wm.tn.task:getTasks	Retrieves all the tasks for this server (the server on which the wm.tn.task:getTasks service is being executed) regardless of the delivery status.
wm.tn.task:getTaskStatus	Retrieves the status of a task on this server (the server on which the wm.tn.task:getTaskStatus service is being executed).
wm.tn.task:reassign	Moves all tasks that have not yet completed from one server to another when using Trading Networks is in a clustered server environment.
wm.tn.task:removeTask	Deletes the specified task.
wm.tn.task:restartTask	Restarts a failed or stopped task.
wm.tn.task:shutdown	Shuts down the task engine.
wm.tn.task:stopTask	Stops the specified task.
wm.tn.task:updateProperties	Allows updates at run-time to parameters used by the task engine.

wm.tn.task:changeServerForTask

Moves a task from one server to another when using Trading Networks in a clustered server environment.

Input Parameters

taskId	**String** The internal identifier for the task to move.
serverId	**String** Host name of the server to which to move the task.

Output Parameters

None

Usage Notes

- If *taskId* is not valid, the service throws an exception. For more information about reassigning tasks to another server, see *webMethods Trading Networks User's Guide*.
- The wm.tn.task:changeServerForTask service invokes the wm.server:connect and wm.server:ping services. As installed, wm.server:connect and wm.server:ping are protected by the Anonymous ACL. To update the Anonymous ACL or use a different ACL to protect the wm.server:connect and wm.server:ping services, the wm.tn.task:changeServerForTask service may be unable to reassign the task.

wm.tn.task:getAnyTask

Retrieves information about a task. If your Integration Server is in a clustered environment, this service can retrieve a task associated with any server in the cluster.

Input Parameters

taskId	**String** The internal identifier for the task to retrieve.
content	**String** Whether to retrieve the content of the document that is associated with the task ID. Valid values are: - `true` - Retrieves document content. - `false` - Default. Does not retrieve document content.
includeBizDocErrors	**String** Prevents the Envelope (see "wm.tn.rec:BizDocEnvelope" on page 331) associated with the task from having the Errors ("wm.tn.rec:BizDocErrorSet" on page 334) field populated with Activity Log error messages. Valid values are: - `true` - Default. Envelope is loaded with errors from the Activity Log. - `false` - Envelope is not loaded with errors from the Activity Log.

Output Parameters

task	**Document** The task identified by *taskId* that this service retrieved. For the structure of *task*, see wm.tn.rec:Task.

Usage Notes

- If *taskId* is not valid, the service throws an exception.
- This service retrieves any task regardless of the server on which the task was started. This service may run slower than wm.tn.task:getTask if it requires information from the Trading Networks database.
- If you do not need the document content, set *content* to `false` to decrease retrieval time.
- If you do not need errors in Envelope, set the *includeBizDocErrors* parameter to false to decrease the retrieval time and memory usage.

wm.tn.task:getTask

Retrieves a specified task that is associated with this server (the server on which the wm.tn.task:getTask service executes).

Input Parameters

taskId	**String** The internal identifier for the task to retrieve.
content	**String** Whether to retrieve the content of the document that is associated with the task. Valid values are: - `true` - Retrieves document content. - `false` - Default. Does not retrieve document content.
includeBizDocErrors	**String** Prevents the Envelope (see "wm.tn.rec:BizDocEnvelope" on page 331) associated with the task from having the Errors ("wm.tn.rec:BizDocErrorSet" on page 334) field populated with Activity Log error messages. Valid values are: - `true` - Default. Envelope is loaded with errors from the Activity Log. - `false` - Envelope is not loaded with errors from the Activity Log.

Output Parameters

task	**Document** The task identified by *taskId* that this service retrieved. For the structure of *task*, see wm.tn.rec:Task.

Usage Notes

- If *taskId* is not valid, the service throws an exception.
- Each task is associated with a single server, so *taskId* must be associated with the same server on which the task was started.
- To get a task that was started on another server, see the wm.tn.task:getAnyTask service.
- If you do not need the document content, set *content* to `false` to decrease retrieval time.
- If you do not need errors in Envelope, set the *includeBizDocErrors* parameter to false to decrease the retrieval time and memory usage.

wm.tn.task:getTaskOutput

Retrieves the output of a specified task.

Input Parameters

taskId	**String** The internal identifier of the task for which you want to retrieve output.
timeout	**String** (optional) The amount of time to wait for output from the task. Specify *timeout* in milliseconds. The default is `0`.

Output Parameters

taskOutput	**Document** The output from the task. For the structure of *taskOutput*, see wm.tn.rec:ReliableServiceOutput.

Usage Notes

- To synchronously retrieve the output from the task, specify a value for *timeout* greater than `0`. When you specify a value for *timeout* that is greater than `0`, a block occurs until the task completes. If the task does not complete in the time you specify, the service throws an exception.
- To check the status of a task before you use this service, call wm.tn.task:getTaskStatus.

wm.tn.task:getTasks

Retrieves all the tasks for this server (the server on which the wm.tn.task:getTasks service executes) regardless of the delivery status.

Input Parameters

None.

Output Parameters

tasks	**String List** The internal identifiers of all tasks.

wm.tn.task:getTaskStatus

Retrieves the status of a task on this server (the server on which the wm.tn.task:getTaskStatus service is being executed).

Input Parameters

taskId	**String** The internal identifier of the task status to retrieve.

Output Parameters

taskStatus	**String** The status of the task. Valid values of *taskStatus* are: ■ `NEW` ■ `PENDING` ■ `DONE` ■ `STOPPED` ■ `FAILED`
taskStatusMsg	**String** The status message of the task.

Usage Notes

If *taskId* is not valid, the service throws an exception.

wm.tn.task:reassign

Moves all tasks that have not yet completed from one server to another when using Trading Networks is in a clustered server environment.

Input Parameters

fromServer	**String** Host name of the server from which to move tasks.
toServer	**String** Host name of the server to which to move tasks.
includeFailed	**String** (optional) Whether to include failed tasks with those tasks that are reassigned. Valid values are: - `true` - Include failed tasks with those that are reassigned. That is, tasks with any of these status values are reassigned: NEW, PENDING, STOPPED, QUEUED, DELIVERING or FAILED. - `false` - Default. Do not include failed tasks with those tasks that are reassigned.

Output Parameters

count	**String** The number of tasks that were reassigned.

Usage Notes

- The value of the *fromServer* input variable should exactly match the value of the **ServerId** field of the tasks to reassign. View the task from My webMethods to verify this value.
- The value of the *toServer* input variable should exactly match the value of the **ServerId** field of tasks on the target server. View the task from My webMethods to verify this value.

wm.tn.task:removeTask

Deletes the specified task.

Input Parameters

taskId	**String** The internal identifier of the task that you want to delete.

Output Parameters

None.

Usage Notes

If *taskId* is not valid, this service throws an exception.

wm.tn.task:restartTask

Restarts a failed or stopped task.

Input Parameters

taskId	**String** The internal identifier of the task to restart.

Output Parameters

None.

Usage Notes

- If *taskId* is not valid, this service throws an exception.
- If the status of the specified task is not "FAILED" or "STOPPED," this service throws an exception.
- Each task is associated with a single server. *taskId* must be associated with the same server on which the task was started.
- You can only restart a failed or stopped task on the same server on which it was started.
- To move a task to another server, use the wm.tn.task:changeServerForTask service.
- This service is set to disable service redirection. For more information, see pub.cluster:disableServiceRedir in *webMethods Integration Server Built-In Services Reference*.

wm.tn.task:shutdown

Shuts down the task engine.

Input Parameters

force	**String** Whether the service shuts down the task engine even if there are tasks pending. Valid values are: - `true` Shuts down the task engine even if there are pending tasks. - `false` Shuts down the task engine only if there are no pending tasks. If the task engine shuts down, the service returns without an exception. If the service was unable to shut down the task engine, the service returns with an exception.

Output Parameters

None.

Usage Notes

If there are tasks pending and *force* is `false`, this service throws an exception.

wm.tn.task:stopTask

Stops the specified task. The task's status is changed to STOP and no more retries are attempted.

Input Parameters

taskId	**String** The internal identifier of the task that you want to stop.

Output Parameters

None.

Usage Notes

- If *taskId* is not valid, this service throws an exception.
- If the task has already completed or "STOPPED," this service throws an exception.

wm.tn.task:updateProperties

Allows updates at run-time to parameters used by the task engine.

Input Parameters

sweepTime	**String** Number of seconds the task engine thread remains idle before checking for tasks to perform (for example, documents it needs to redeliver or services it needs to execute). For more information, see the description of the Trading Networks property `tn.task.sweepTime` in *webMethods Trading Networks Administrator's Guide*.

Output Parameters

None.

19 TPA Folder

Overview

Use the services in the wm.tn.tpa folder to create and manage Trading Partners Agreements (TPAs).

Summary of Elements in this Folder

The following table describes the elements that are available in this folder:

Element	Description
wm.tn.tpa:changeStatus	Changes the status of the TPA.
wm.tn.tpa:createTPA	Creates a TPA.
wm.tn.tpa:deleteTPA	Deletes a TPA.
wm.tn.tpa:getTPA	Retrieves a TPA.
wm.tn.tpa:getTPAInLock	Retrieves a TPA within a locking-block.
wm.tn.tpa:getTPALock	Requests a lock for the TPA that matches the given *senderID*, *receiverID*, and *agreementID*.
wm.tn.tpa:nextControlNumber	Increases the value of *controlNumber* by one, and returns the new value of *controlNumber*.
wm.tn.tpa:releaseTPALock	Releases a lock.
wm.tn.tpa:setLockError	Sets an error condition on a lock.
wm.tn.tpa:updateControlNumber	Updates the value of *controlNumber*.
wm.tn.tpa:updateControlNumberInLock	Updates the value of *controlNumber* within a locking-block.
wm.tn.tpa:updateTPA	Updates a TPA.
wm.tn.tpa:updateTPAData	Updates the data in a TPA.
wm.tn.tpa:updateTPADataInLock	Updates the TPA data in a TPA within a locking-block.
wm.tn.tpa:validateTPA	Validates the TPA data against the TPA data schema.

wm.tn.tpa:changeStatus

Changes the status of the TPA.

Input Parameters

senderID	**String** The internal ID of the trading partner that has the sender role in the transaction the TPA governs.
receiverID	**String** The internal ID of the trading partner that has the receiver role in the transaction the TPA governs.
agreementID	**String** The agreement ID of the TPA.
newStatus	**String** The new status of the TPA. It can have three values: `proposed`, `agreed`, or `disabled`.

Output Parameters

error	**Document** (optional) The error reported by the service.

wm.tn.tpa:createTPA

Creates a TPA. If the specified *senderID*, *receiverID*, and *agreementID* are not unique, the service reports an error. If the service fails to create a TPA for other reasons, it throws an exception. If you specify *initService* but not *tpaData*, the service generates the default value for *tpaData*.

Input Parameters

senderID	**String** The internal ID of the trading partner that has the sender role in the transaction the TPA governs.
receiverID	**String** The internal ID of the trading partner that has the receiver role in the transaction the TPA governs.
agreementID	**String** The agreement ID of the TPA.
status	**String** The status of the TPA. It can have one of three values: `proposed`, `disabled`, or `agreed`.
controlNumber	**String** (optional) A placeholder for an integer. Its usage is application-specific.
exportService	**String** (optional) The fully-qualified service name of a service that you provide to convert TPA data to a format that can be exported to non-Trading Networks systems.
initService	**String** (optional) A fully-qualified service name of a service that you provide to populate the TPA data (*tpaData*) with default values.
dataStatus	**String** (optional) Whether the data in *tpaData* is modifiable. Valid values are:

- `mutable` - *tpaData* is modifiable.
- `immutable` - *tpaData* is not modifiable.

dataSchema	**String** (optional) A blueprint of the TPA that establishes the TPA parameters and values.
tpaData	**Document** (optional) TPA data.

Output Parameters

tpa	**Document** The TPA object.
error	**Document** (optional) The error reported by the service.

wm.tn.tpa:deleteTPA

Deletes a TPA. If the TPA does not exist or if the TPA status is "agreed," the service reports an error. For other service invocation- or database-related errors, it throws an exception.

Input Parameters

tpaID	**String** (optional) The internal ID of the trading partner agreement. If you do not specify *tpaID*, you must specify *senderID*, *receiverID* and *agreementID*.
senderID	**String** The internal ID of the trading partner that has the sender role in the transaction the TPA governs. If *tpaID* is not specified, then you must specify *senderID*, *receiverID* and *agreementID*.
receiverID	**String** The internal ID of the trading partner that has the receiver role in the transaction the TPA governs. If *tpaID* is not specified, then you must specify *senderID*, *receiverID* and *agreementID*.
agreementID	**String** The agreement ID of the TPA. If *tpaID* is not specified, then you must specify *senderID*, *receiverID* and *agreementID*.

Output Parameters

error	**Document** (optional) The error reported by the service.

wm.tn.tpa:getTPA

Retrieves a TPA. If no TPA is found, the service returns null. For other database or service invocation-related errors, it throws an exception.

Input Parameters

senderID	**String** The internal ID of the trading partner that has the sender role in the transaction the TPA governs. To specify the unknown partner, use `Unknown`.
receiverID	**String** The internal ID of the trading partner that has the receiver role in the transaction the TPA governs. To specify the unknown partner, use `Unknown`.
agreementID	**String** The agreement ID of the TPA.
direction	**String** (optional) Indicates whether the service searches for a TPA where the sender and receiver are switched, if a TPA with the sender/receiver does not exist. Valid values are: ■ `true` - The service only looks for a TPA where the sender/receiver are as specified. If not, no TPA is returned. ■ `false` - If a TPA with the specified sender/receiver does not exist, the service switches the specified sender/receiver and: 1. Searches for a TPA where the sender matches the specified receiver and the receiver matches the specified sender. 2. Returns the TPA if found.

Output Parameters

tpa	**Document** (optional) The retrieved TPA as an IData object that has the format defined by the wm.tn.rec:tpa IS document type.
error	**Document** The error reported by the service.

wm.tn.tpa:getTPAInLock

Retrieves a TPA within a locking-block. For other service invocation- or database-related errors, the service throws an exception.

Input Parameters

lock	**Object** The lock object.

Output Parameters

tpa	**Document** (optional) The retrieved TPA as an IData object that has the format defined by the wm.tn.rec:tpa IS document type.
error	**Document** The error reported by the service.

wm.tn.tpa:getTPALock

Requests a lock for the TPA that matches the given *senderID*, *receiverID*, and *agreementID*.

The service blocks until a lock is available. If the TPA does not exist, the service reports an error. For other service invocation- or database-related errors, it throws an exception. When an application is done with a lock, it must release the lock using wm.tn.tpa:releaseTPALock. You can use only the following services within a locking block:

- wm.tn.tpa:getTPAInLock
- wm.tn.tpa:updateControlNumberInLock
- wm.tn.tpa:updateTPADataInLock
- wm.tn.tpa:setLockError

Input Parameters

senderID	**String** The internal ID of the trading partner that has the sender role in the transaction the TPA governs.
receiverID	**String** The internal ID of the trading partner that has the receiver role in the transaction the TPA governs.
agreementID	**String** The agreement ID of the TPA.

Output Parameters

lock	**Object** The lock object returned by the service.
error	**Document List** The error reported by the service. If no error is found, this is null.

wm.tn.tpa:nextControlNumber

Increases the value of *controlNumber* by one, and returns the new value of *controlNumber*. If the service does not find a TPA, it reports an error.

Input Parameters

senderID	**String** The internal ID of the trading partner that has the sender role in the transaction the TPA governs.
receiverID	**String** The internal ID of the trading partner that has the receiver role in the transaction the TPA governs.
agreementID	**String** The agreement ID of the TPA.

Output Parameters

controlNumber	**String** The new *controlNumber*.
error	**Document** (optional) The error reported by the service.

wm.tn.tpa:releaseTPALock

Releases a lock. An application must release a lock after it has finished with the lock. If an application does not release a lock after it has finished with the lock, the lock is released when it is garbage-collected.

Input Parameters

lock	**String** The lock object that is released.

Output Parameters

None.

wm.tn.tpa:setLockError

Sets an error condition on a lock. Upon release of the lock, all database updates during the locking-block are rolled back.

Input Parameters

lock	**Object** The lock object.
error	**Document** An error IData object.

Output Parameters

None.

wm.tn.tpa:updateControlNumber

Updates the value of *controlNumber*. If the TPA does not exist or if the specified value of *newControlNumber* is not an integer, the service reports an error. For other service invocation- or database-related errors, it throws an exception.

Input Parameters

senderID	**String** The internal ID of the trading partner that has the sender role in the transaction the TPA governs.
receiverID	**String** The internal ID of the trading partner that has the receiver role in the transaction the TPA governs.
agreementID	**String** The agreement ID of the TPA.
newControlNumber	**String** The new control number.

Output Parameters

error	**Document** (optional) The error reported by the service.

wm.tn.tpa:updateControlNumberInLock

Updates the value of *controlNumber* within a locking-block.

If the specified value of *newControlNumber* is not an integer, the service reports an error.

Input Parameters

lock	**Object** The lock object.
newControlNumber	**String** The new controlNumber.

Output Parameters

error	**Document** (optional) The error reported by the service.

wm.tn.tpa:updateTPA

Updates a TPA. If the TPA does not exist or if the TPA status is “agreed,” the service reports an error. For other service invocation- or database-related errors, it throws an exception.

Input Parameters

senderID	**String** The internal ID of the trading partner that has the sender role in the transaction the TPA governs.
receiverID	**String** The internal ID of the trading partner that has the receiver role in the transaction the TPA governs.
agreementID	**String** The agreement ID of the TPA.
status	**String** (optional) The status of the TPA. It can have one of three values: `proposed`, `disabled`, or `agreed`.
controlNumber	**String** (optional) A placeholder for an integer. Its usage is application-specific.
exportService	**String** (optional) The fully-qualified service name of a service that you provide to convert TPA data to a format that can be exported to non-Trading Networks systems.
initService	**String** (optional) A fully-qualified service name of a service that you provide to populate the TPA data (*tpaData*) with default values.
dataStatus	**String** (optional) Whether the data in tpaData is modifiable. Valid values are: ■ `mutable` - *tpaData* is modifiable. ■ `immutable` - *tpaData* is not modifiable.
dataSchema	**String** (optional) A blueprint of the TPA that establishes the TPA parameters and values.
tpaData	**Document** (optional) TPA data.

Output Parameters

error	**Document** (optional) The error reported by the service.

wm.tn.tpa:updateTPAData

Updates the data in a TPA. If the service does not find a TPA or if the TPA is mutable, the service reports an error.

Input Parameters

senderID	**String** The internal ID of the trading partner that has the sender role in the transaction the TPA governs.

receiverID	**String** The internal ID of the trading partner that has the receiver role in the transaction the TPA governs.
agreementID	**String** The agreement ID of the TPA.
newData	**Document** The new TPA data.

Output Parameters

error	**Document** (optional) The error reported by the service.

wm.tn.tpa:updateTPADataInLock

Updates the TPA data in a TPA within a locking-block. If the TPA data is not mutable, the service reports an error.

Input Parameters

lock	**Object** The lock object.
newData	**Document** The updated TPA data.

Output Parameters

error	**Document** (optional) An error IData object.

wm.tn.tpa:validateTPA

Validates the TPA data against the TPA data schema.

Input Parameters

senderID	**String** The internal ID of the trading partner that has the sender role in the transaction the TPA governs.
receiverID	**String** The internal ID of the trading partner that has the receiver role in the transaction the TPA governs.
agreementID	**String** The agreement ID of the TPA.

Output Parameters

error	**Document List** (optional) The error reported by the service. If no error is found, this is null.

20 Transport Folder

Overview

Use the transport services (services in the wm.tn.transport folder) to deliver outbound documents using the various transport protocols.

These transport services deliver the document that is an instance of com.wm.app.tn.doc.BizDocEnvelope. A BizDocEnvelope can have multiple content parts, but these services deliver only that content part, which is considered as delivery content for the specified document type. The delivery content differs for different document types. See the respective e-standards documentation to find the delivery content of the corresponding document type. In Trading Networks the built-in document types are XML and Flatfile, and the delivery contents are xmldata and ffdata content parts respectively.

Each service in the wm.tn.transport folder represents an immediate delivery method that you create in Trading Networks, the built-in immediate delivery method, or a scheduled delivery service that Trading Networks provides.

At run-time, the wm.tn.transport services used for immediate delivery methods require additional information (for example, host name and port). To obtain this information, the transport service determines the receiver of the document it is transporting. It then looks up the receiver's profile to find the specific delivery method parameters it should use.

For wm.tn.transport services used for scheduled delivery, the transport service obtains the information it needs at run-time from the pipeline.

Summary of Elements in this Folder

The following table describes the elements that are available in this folder:

Element	Description
wm.tn.transport:activeTransfer	Delivers documents using webMethods ActiveTransfer.
wm.tn.transport:batchFtp	Delivers all documents on the specified scheduled delivery queue using FTP.
wm.tn.transport:Ftp	Delivers documents through FTP.
wm.tn.transport:Ftps	Delivers documents through FTP over SSL (also known as FTPS).
wm.tn.transport:Sftp	Delivers documents through SFTP.
wm.tn.transport:Http	Delivers documents through HTTP.
wm.tn.transport:Https	Delivers documents through HTTPS.
wm.tn.transport:primaryFtp	Delivers documents through FTP.

Element	Description
wm.tn.transport:primaryFtps	Delivers documents through FTP over SSL (also known as FTPS).
wm.tn.transport:primaryHttp	Delivers documents through HTTP.
wm.tn.transport:primaryHttps	Delivers documents through HTTPS.
wm.tn.transport:primarySmtp	Delivers documents through SMTP.
wm.tn.transport:primarySftp	Delivers documents through SFTP.
wm.tn.transport:secondaryFtp	Delivers documents through FTP.
wm.tn.transport:secondaryFtps	Delivers documents through FTP.
wm.tn.transport:secondaryHttp	Delivers documents through HTTP.
wm.tn.transport:secondaryHttps	Delivers documents through HTTPS.
wm.tn.transport:secondarySmtp	Delivers documents through SMTP.
wm.tn.transport:Smtp	Delivers documents through SMTP.
wm.tn.transport:secondarySftp	Delivers documents through SFTP.
wm.tn.transport:webService	Delivers documents through a Web service.

wm.tn.transport:activeTransfer

Delivers documents using ActiveTransfer. This service uses the delivery method information from the receiver's partner profile to transport documents through ActiveTransfer to the partner's endpoint (Virtual Folder System or VFS). The information determines the *vfsId*, *vfsPath*, *senderPartnerId*, *receiverPartnerId*, and *documentBytes* of the document to deliver.

Note:
The wm.tn.transport:activeTransfer service does not support the delivery of large documents to ActiveTransfer on a remote Integration Server instance. For more information about handling large documents in Trading Networks, see the 'Large Document Handling' chapter in *webMethods Trading Networks Administrator's Guide*.

Input Parameters

serviceName	**String**: The name of the service you must use to deliver a document using ActiveTransfer.
bizdoc	**Document**: The document to deliver. wm.tn.rec:BizDocEnvelope defines the structure of the bizdoc.

retriesRemaining **String**: The number of pending retries. The delivery engine passes this information to the server at every attempt of the job delivery.

Output Parameters

serviceOutput **Document** The output from the delivery. For the structure of *serviceOutput*, see "wm.tn.rec:activeTransferOutput" on page 343. The document contains the following keys

- *status* **String**: The outcome of the delivery: `success` or `fail`.
- *statusMessage* **String**: The status message from the last attempt to deliver the document.

  In case of `success`, an output message is displayed. In case of `fail`, an exception message is displayed.
- *vfsPath* **String**: ActiveTransfer VFS path to which the document is delivered.
- *transportTime* **String**: Total time (in milliseconds) it took to deliver the document.
- *output* **Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:activeTransferOutput.
  - *success* **String** `true` If the Trading Networks document is successfully delivered to ActiveTransfer or `false`: If the Trading Networks document is successfully delivered to ActiveTransfer.
  - *transactionId* **String**: A unique ID returned by ActiveTransfer to identify the document transferred from Trading Networks to ActiveTransfer.
  - *message* **String** A message that is logged after the service execution is complete.

wm.tn.transport:batchFtp

Delivers all documents on the specified scheduled delivery queue using FTP. This service is provided as a reference implementation that you can use as a basis for your own scheduled delivery service. For more information about creating scheduled delivery services, see *webMethods Trading Networks Administrator's Guide*.

Input Parameters

queue **String** The name of the scheduled delivery queue from which to deliver documents.

host	**String** The host name or IP address of the target FTP server.
port	**String** The port number on which the target FTP server listens for requests. The default is 21.
user	**String** (optional) The name of the account to log on the target FTP server.
password	**String** (optional) The password of the account to log on the target FTP server.
directory	**String** (optional) The directory on the target FTP server in which you want the documents written.
transfermode	**String** The FTP data transfer mode: either `ascii` or `binary`.
transfertype	**String** The FTP data transfer type: either `active` or `passive`.
fileExtension	**String** (optional) The extension to use for target file names. If you do not specify a value for *fileExtension*, this service uses the value returned from BizDocType.getFtpFileExtension. For XMLDocTypes, this value is `xml`. The target file name is: task/Envelope/*InternalID.fileExtension*
dataport	**String** (optional) The listener port number of the data transfer channel. If you do not specify *dataport*, Trading Networks chooses the listener port number. This service only uses *dataport* when you specify `active` for *transfertype*.
encoding	**String** (optional) Character set in which the document is encoded. This variable is required to convert the String object to bytes correctly. Specify an IANA-registered character set. If this variable is null, the default JVM encoding is used. *Example:* ISO-8859-1
timeout	**String** (optional) Number of seconds to wait for a response from the ftp server before timing out and aborting the request. The default is to wait indefinitely.

Output Parameters

logMsg	**String** FTP log messages for the entire user session.

Usage Notes

You create a scheduled delivery queue using My webMethods. If you select **Batch FTP** as the delivery service for the queue, Trading Networks invokes this service to deliver documents from the queue. When you define or update the settings for the scheduled delivery queue, you can

supply values for the service inputs. The procedures for defining, updating, and managing schedule delivery queues are described in *webMethods Trading Networks Administrator's Guide*.

You should not invoke this service directly. If you want to programmatically deliver documents from a queue, invoke the wm.tn.queuing:deliverBatch service and supply the queue name.

wm.tn.transport:Ftp

Delivers documents through FTP. This service uses the delivery method information from the receiving partner's profile for the delivery method that uses FTP. The information determines the host name, port number, user name, password, and directory to use to deliver the document.

Input Parameters

serviceName	**String** The name of the service you must use to deliver a document using FTP.
bizdoc	**Document** The document you want to deliver. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
retriesRemaining	**String** The number of retries remaining. The delivery engine passes this information into the server at every attempt of job delivery.

Output Parameters

serviceOutput	**Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:FtpDeliveryServiceOutput. The document contains the following keys: ■ *status* **String** The outcome of the delivery: `success` or `fail`. ■ *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be `221`. ■ *transportTime* **String** Total time (in milliseconds) it took to deliver the document. ■ *output* **Document** The header and body that was returned from the FTP "put". For the structure of *output*, see wm.tn.rec:FtpOutput.

a

Usage Notes

- The service throws an exception if the receiver identified to receive the document in `bizdoc` is a local partner. And the local partner is a host of the network.
- The service performs a passive transfer.

- The file name of the delivered document is the internal document ID and the extension is either .xml if the content-type is text, xml, or `.bizdoc`.

wm.tn.transport:Ftps

Delivers documents through FTP over SSL (also known as FTPS). This service uses the delivery method information from the receiving partner's profile for the delivery method that uses FTPS. The information determines the host name, port number, user name, password, and directory to use to deliver the document.

Input Parameters

serviceName	**String** The name of the service you must use to deliver a document using FTPS.
bizdoc	**Document** The document you want to deliver. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
retriesRemaining	**String** The number of retries remaining. The delivery engine passes this information into the server at every attempt of job delivery.

Output Parameters

serviceOutput — **Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:FtpDeliveryServiceOutput. The document contains the following keys:

- *status* **String** The outcome of the delivery: `success` or `fail`.
- *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be `221`.
- *transportTime* **String** Total time (in milliseconds) it took to deliver the document.
- *output* **Document** The header and body that was returned from the FTP `put`. For the structure of *output*, see wm.tn.rec:FtpOutput.

Usage Notes

- This service can establish connections only to FTP servers that use SSL; it cannot establish connections to servers that implement other security facilities.
- FTPS requires an SSL handshake, which results in additional processing and additional exposure to network latency. If very large documents are being transmitted, this additional overhead is likely to be negligible. However, if FTPS is used to deliver small documents, the additional overhead may be significant.

- The service throws an exception if the receiver identified to receive the document in `bizdoc` is a local partner. And the local partner is a host of the network.
- This service performs a passive transfer.
- The file name of the delivered document is the internal document ID and the extension is either .xml if the content-type is text, xml, or `.bizdoc` .

wm.tn.transport:Sftp

Delivers documents through SFTP. This service uses the delivery method information from the receiving partner's profile to determine the SFTP user alias and the location to which the document must be delivered.

Input Parameters

serviceName	**String** The name of the service you must use to deliver a document using SFTP.
bizdoc	**Document** The document you want to deliver. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
retriesRemaining	**String** The number of retries remaining. The delivery engine passes this information into the server at every attempt of job delivery.

Output Parameters

serviceOutput	**Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:SftpDeliveryServiceOutput. The document contains the following keys: - *status* **String** The outcome of the delivery: `success` or `fail`. - *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be `221`. - *transportTime* **String** Total time (in milliseconds) it took to deliver the document. - *output* **Document** The header and body that was returned from the SFTP `put`. For the structure of *output*, see wm.tn.rec:SftpOutput.

Usage Notes

- The service throws an exception if the receiver identified to receive the document in `bizdoc` is a local partner. And the local partner is a host of the network.
- The service performs a passive transfer.

- The file name of the delivered document is the internal document ID and the extension is either .xml if the content-type is text, xml, or `.bizdoc`.

wm.tn.transport:Http

Delivers documents through HTTP. This service uses the delivery method information from the receiving partner's profile for the delivery method that uses HTTP. The information determines the host name, port number, user name, password, and URL to use to deliver the document.

Input Parameters

serviceName	**String** The name of the service you must use to deliver a document using HTTPS .
bizdoc	**Document** The document you want to deliver. If invoking from a Java program, the document must be an instance of com.wm.app.tn.doc.BizDocEnvelope. Otherwise, the structure of *bizdoc* is defined by wm.tn.rec:BizDocEnvelope.
retriesRemaining	**String** The number of retries remaining. The delivery engine passes this information into the server at every attempt of job delivery.

Output Parameters

serviceOutput	**Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:HttpDeliveryServiceOutput. The document contains the following keys: - *status* **String** The outcome of the delivery: `success` or `fail`. - *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be `200 OK`. - *transportTime* **String** Total time (in milliseconds) it took to deliver the document. - *output* **Document** The header and body that was returned from the HTTP post. For the structure of *output*, see wm.tn.rec:HttpOutput.

Usage Notes

The service throws an exception if the receiver identified to receive the document in is a local partner. And the local partner is a host of the network.

wm.tn.transport:Https

Delivers documents through HTTPS. This service uses the delivery method information from the receiving partner's profile for the delivery method that uses HTTPS. The information determines the host name, port number, user name, password, and URL to use to deliver the document.

Input Parameters

serviceName **String** The name of the service you must use to deliver a document using HTTPS.

bizdoc **Document** The document you want to deliver.

If invoking from a Java program, the document must be an instance of com.wm.app.tn.doc.BizDocEnvelope. Otherwise, the structure of *bizdoc* is defined by wm.tn.rec:BizDocEnvelope.

retriesRemaining **String** The number of retries remaining. The delivery engine passes this information into the server at every attempt of job delivery.

Output Parameters

serviceOutput **Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:HttpDeliveryServiceOutput. The document contains the following keys:

- *status* **String** The outcome of the delivery: either `success` or `fail`.
- *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be `200 OK`.
- *transportTime* **String** Total time (in milliseconds) it took to deliver the document.
- *output* **Document** The header and body that was returned from the HTTP post. For the structure of *output*, see wm.tn.rec:HttpOutput.

Usage Notes

The service throws an exception if the receiver identified to receive the document in `bizdoc` is a local partner. And the local partner is a host of the network.

wm.tn.transport:primaryFtp

Delivers documents through FTP. This service uses the delivery method information that is associated with the primary FTP delivery method from the receiving partner's profile to determine the host name, port number, user name, password, and directory to use to deliver the document.

Input Parameters

bizdoc	**Document** The document you want to deliver. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
retriesRemaining	**String** The number of retries remaining. The delivery engine passes this information into this server at every attempt of job delivery.

Output Parameters

serviceOutput **Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:FtpDeliveryServiceOutput. The document contains the following keys:

- *status* **String** The outcome of the delivery: `success` or `fail`.
- *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be `221`.
- *transportTime* **String** Total time (in milliseconds) it took to deliver the document.
- *output* **Document** The header and body that was returned from the FTP `put`. For the structure of *output*, see wm.tn.rec:FtpOutput.

Usage Notes

- The service throws an exception if the receiver identified to receive the document in `bizdoc` is a local partner. And the local partner is a host of the network.
- This service performs a passive transfer.
- The file name of the delivered document is the internal document ID and the extension is either .xml if the content-type is text, xml, or `.bizdoc`.

wm.tn.transport:primaryFtps

Delivers documents through FTP over SSL (also known as FTPS). This service uses the delivery method information that is associated with the primary FTPS delivery method from the receiving partner's profile to determine the host name, port number, user name, password, and directory to use to deliver the document.

Input Parameters

bizdoc	**Document** The document you want to deliver. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
retriesRemaining	**String** The number of retries remaining. The delivery engine passes this information into this server at every attempt of job delivery.

Output Parameters

serviceOutput **Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:FtpDeliveryServiceOutput. The document contains the following keys:

- *status* **String** The outcome of the delivery: `success` or `fail`.
- *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be `221`.
- *transportTime* **String** Total time (in milliseconds) it took to deliver the document.
- *output* **Document** The header and body that was returned from the FTP `put`. For the structure of *output*, see wm.tn.rec:FtpOutput.

Usage Notes

- This service can establish connections only to FTP servers that use SSL; it *cannot* establish connections to servers that implement other security facilities.
- FTPS requires an SSL handshake, which results in additional processing and additional exposure to network latency. If very large documents are being transmitted, this additional overhead is likely to be negligible. However, if FTPS is used to deliver small documents, the additional overhead may be significant.
- The service throws an exception if the receiver identified to receive the document in `bizdoc` is a local partner. And the local partner is a host of the network.
- This service performs a passive transfer.
- The file name of the delivered document is the internal document ID and the extension is either .xml if the content-type is text, xml, or `.bizdoc`.

wm.tn.transport:primaryHttp

Delivers documents through HTTP. This service uses the delivery method information that is associated with the primary HTTP delivery method from the receiving partner's profile to determine the host name, port number, user name, password, and URL to use to deliver the document.

Input Parameters

bizdoc **Document** The document you want to deliver.

If invoking from a Java program, the document must be an instance of com.wm.app.tn.doc.BizDocEnvelope. Otherwise, the structure of *bizdoc* is defined by wm.tn.rec:BizDocEnvelope.

retriesRemaining **String** The number of retries remaining. The delivery engine passes this information into this server at every attempt of job delivery.

Output Parameters

serviceOutput **Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:HttpDeliveryServiceOutput. The document contains the following keys:

- *status* **String** The outcome of the delivery: `success` or `fail`.
- *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be `200 OK`.
- *transportTime* **String** Total time (in milliseconds) it took to deliver the document.
- *output* **Document** The header and body that was returned from the HTTP post. For the structure of *output*, see wm.tn.rec:HttpOutput.

Usage Notes

If the receiver identified in *bizdoc* (the partner to receive the document being delivered) is the local partner (the host of the network), this service throws an exception.

wm.tn.transport:primaryHttps

Delivers documents through HTTPS. This service uses the delivery method information that is associated with the primary HTTPS delivery method from the receiving partner's profile to determine the host name, port number, user name, password, and URL to use to deliver the document.

Input Parameters

bizdoc **Document** The document you want to deliver.

If invoking from a Java program, the document must be an instance of com.wm.app.tn.doc.BizDocEnvelope. Otherwise, the structure of *bizdoc* is defined by wm.tn.rec:BizDocEnvelope.

retriesRemaining **String** The number of retries remaining. The delivery engine passes this information into this server at every attempt of job delivery.

Output Parameters

serviceOutput **Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:HttpDeliveryServiceOutput. The document contains the following keys:

- *status* **String** The outcome of the delivery: `success` or `fail`.
- *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be `200 OK`.
- *transportTime* **String** Total time (in milliseconds) it took to deliver the document.
- *output* **Document** The header and body that was returned from the HTTP post. For the structure of *output*, see wm.tn.rec:HttpOutput.

Usage Notes

If the receiver identified in *bizdoc* (the partner to receive the document being delivered) is the local partner (the host of the network), this service throws an exception.

wm.tn.transport:primarySmtp

Delivers documents through SMTP. This service uses the delivery method information that is associated with the primary email delivery method from the receiving partner's profile to determine the email address to use to deliver the document.

Input Parameters

bizdoc **Document** The document you want to deliver.

If invoking from a Java program, the document must be an instance of com.wm.app.tn.doc.BizDocEnvelope. Otherwise, the structure of *bizdoc* is defined by wm.tn.rec:BizDocEnvelope.

retriesRemaining **String** The number of retries remaining. The delivery engine passes this information into this server at every attempt of job delivery.

Output Parameters

serviceOutput **Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:SmtpDeliveryServiceOutput. The document contains the following keys:

- *status* **String** The outcome of the delivery: `success` or `fail`.
- *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be Mail was sent successfully.
- *transportTime* **String** Total time (in milliseconds) it took to deliver the document.
- *output* **Document** For the structure of *output*, see wm.tn.rec:SmtpOutput.

Usage Notes

The service throws an exception if the receiver identified to receive the document in `bizdoc` is a local partner. And the local partner is a host of the network.

wm.tn.transport:primarySftp

Delivers documents through SFTP. This service uses the delivery method information that is associated with the primary SFTP delivery method from the receiving partner's profile. This information determines the SFTP user alias and the location to which the document must be delivered.

Input Parameters

bizdoc **Document** The document you want to deliver. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.

retriesRemaining **String** The number of retries remaining. The delivery engine passes this information into this server at every attempt of job delivery.

Output Parameters

serviceOutput **Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:SftpDeliveryServiceOutput. The document contains the following keys:

- *status* **String** The outcome of the delivery: `success` or `fail`.

- *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be `221`.
- *transportTime* **String** Total time (in milliseconds) it took to deliver the document.
- *output* **Document** The header and body that was returned from the SFTP `put`. For the structure of *output*, see wm.tn.rec:SftpOutput.

Usage Notes

- The service throws an exception if the receiver identified to receive the document in `bizdoc` is a local partner. And the local partner is a host of the network.
- This service performs a passive transfer.
- The file name of the delivered document is the internal document ID and the extension is either .xml if the content-type is text, xml, or `.bizdoc`.

wm.tn.transport:secondaryFtp

Delivers documents through FTP. This service uses the delivery method information that is associated with the secondary FTP delivery method from the receiving partner's profile to determine the host name, port number, user name, password, and directory to use to deliver the document.

Input Parameters

bizdoc	**Document** The document you want to deliver. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
retriesRemaining	**String** The number of retries remaining. The delivery engine passes this information into this server at every attempt of job delivery.

Output Parameters

serviceOutput	**Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:FtpDeliveryServiceOutput. The document contains the following keys: - *status* **String** The outcome of the delivery: `success` or `fail`. - *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be `221`. - *transportTime* **String** Total time (in milliseconds) it took to deliver the document.

- *output* **Document** The header and body that was returned from the FTP "put." For the structure of *output*, see wm.tn.rec:FtpOutput.

Usage Notes

- The service throws an exception if the receiver identified to receive the document in `bizdoc` is a local partner. And the local partner is a host of the network.
- This service performs a passive transfer.
- The file name of the delivered document is the internal document ID and the extension is either .xml if the content-type is text, xml, or `.bizdoc` .

wm.tn.transport:secondaryFtps

Delivers documents through FTP over SSL (also known as FTPS). This service uses the delivery method information that is associated with the secondary FTP delivery method from the receiving partner's profile to determine the host name, port number, user name, password, and directory to use to deliver the document.

Input Parameters

bizdoc	**Document** The document you want to deliver. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
retriesRemaining	**String** The number of retries remaining. The delivery engine passes this information into this server at every attempt of job delivery.

Output Parameters

serviceOutput	**Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:FtpDeliveryServiceOutput. The document contains the following keys: - *status* **String** The outcome of the delivery: `success` or `fail`. - *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be `221`. - *transportTime* **String** Total time (in milliseconds) it took to deliver the document. - *output* **Document** The header and body that was returned from the FTP "put." For the structure of *output*, see wm.tn.rec:FtpOutput.

Usage Notes

- This service can establish connections only to FTP servers that use SSL; it *cannot* establish connections to servers that implement other security facilities.
- FTPS requires an SSL handshake, which results in additional processing and additional exposure to network latency. If very large documents are being transmitted, this additional overhead is likely to be negligible. However, if FTPS is used to deliver small documents, the additional overhead may be significant.
- The service throws an exception if the receiver identified to receive the document in `bizdoc` is a local partner. And the local partner is a host of the network.
- This service performs a passive transfer.
- The file name of the delivered document is the internal document ID and the extension is either .xml if the content-type is text, xml, or `.bizdoc`.

wm.tn.transport:secondarySftp

Delivers documents through SFTP. This service uses the delivery method information that is associated with the secondary SFTP delivery method from the receiving partner's profile. The delivery method information determines the SFTP user alias and the location to which the document must be delivered.

Input Parameters

bizdoc	**Document** The document you want to deliver. The document must be an instance of com.wm.app.tn.doc.BizDocEnvelope.
retriesRemaining	**String** The number of retries remaining. The delivery engine passes this information into this server at every attempt of job delivery.

Output Parameters

serviceOutput	**Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:SftpDeliveryServiceOutput. The document contains the following keys: ■ *status* **String** The outcome of the delivery:`success` or `fail`. ■ *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be `221`. ■ *transportTime* **String** Total time (in milliseconds) it took to deliver the document.

- *output* **Document** The header and body that was returned from the FTP "put." For the structure of *output*, see wm.tn.rec:SftpOutput.

Usage Notes

- The service throws an exception if the receiver identified to receive the document in `bizdoc` is a local partner. And the local partner is a host of the network.
- This service performs a passive transfer.
- The file name of the delivered document is the internal document ID and the extension is either .xml if the content-type is text, xml, or `.bizdoc`.

wm.tn.transport:secondaryHttp

Delivers documents through HTTP. This service uses the delivery method information that is associated with the secondary HTTP delivery method from the receiving partner's profile to determine the host name, port number, user name, password, and URL to use to deliver the document.

Input Parameters

bizdoc	**Document** The document you want to deliver. If invoking from a Java program, the document must be an instance of com.wm.app.tn.doc.BizDocEnvelope. Otherwise, the structure of *bizdoc* is defined by wm.tn.rec:BizDocEnvelope.
retriesRemaining	**String** The number of retries remaining. The delivery engine passes this information into this server at every attempt of job delivery.

Output Parameters

serviceOutput	**Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:HttpDeliveryServiceOutput. The document contains the following keys: - *status* **String** The outcome of the delivery: `success` or `fail`. - *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be `200 OK`. - *transportTime* **String** Total time (in milliseconds) it took to deliver the document.

- *output* **Document** The header and body that was returned from the HTTP post. For the structure of *output*, see wm.tn.rec:HttpOutput.

Usage Notes

The service throws an exception if the receiver identified to receive the document in `bizdoc` is a local partner. And the local partner is a host of the network.

wm.tn.transport:secondaryHttps

Delivers documents through HTTPS. This service uses the delivery method information that is associated with the Secondary HTTPS delivery method from the receiving partner's profile to determine the host name, port number, user name, password, and URL to use to deliver the document.

Input Parameters

bizdoc **Document** The document you want to deliver.

If invoking from a Java program, the document must be an instance of com.wm.app.tn.doc.BizDocEnvelope. Otherwise, the structure of *bizdoc* is defined by wm.tn.rec:BizDocEnvelope.

retriesRemaining **String** The number of retries remaining. The delivery engine passes this information into this server at every attempt of job delivery.

Output Parameters

serviceOutput **Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:HttpDeliveryServiceOutput. The document contains the following keys:

- *status* **String** The outcome of the delivery: `success` or `fail`.
- *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be `200 OK`.
- *transportTime* **String** Total time (in milliseconds) it took to deliver the document.
- *output* **Document** The header and body that was returned from the HTTP post. For the structure of *output*, see wm.tn.rec:HttpOutput.

Usage Notes

The service throws an exception if the receiver identified to receive the document in `bizdoc` is a local partner. And the local partner is a host of the network.

wm.tn.transport:secondarySmtp

Delivers documents through SMTP. This service uses the delivery method information that is associated with the secondary email delivery method from the receiving partner's profile to determine the email address to use to deliver the document.

Delivers documents through SMTP using the partner's secondary email address.

Input Parameters

bizdoc	**Document** The document you want to deliver. If invoking from a Java program, the document must be an instance of com.wm.app.tn.doc.BizDocEnvelope. Otherwise, the structure of *bizdoc* is defined by wm.tn.rec:BizDocEnvelope.
retriesRemaining	**String** The number of retries remaining. The delivery engine passes this information into this server at every attempt of job delivery.

Output Parameters

serviceOutput	**Document** The output from the delivery. For the format of *serviceOutput*, see wm.tn.rec:SmtpDeliveryServiceOutput. The document contains the following keys: ■ *status* **String** The outcome of the delivery: `success` or `fail`. ■ *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be "Mail was sent successfully." ■ *transportTime* **String** Total time (in milliseconds) it took to deliver the document. ■ *output* **Document** For the structure of *output*, see wm.tn.rec:SmtpOutput.

Usage Notes

The service throws an exception if the receiver identified to receive the document in `bizdoc` is a local partner. And the local partner is a host of the network.

wm.tn.transport:Smtp

Delivers documents through SMTP. This service uses the delivery method information from the receiving partner's profile for the delivery method that uses email as the delivery method. The information determines the email address to use to deliver the document.

Input Parameters

serviceName **String** The name of the delivery method that must be used to deliver the document through SMTP.

bizdoc **Document** The document you want to deliver.

If invoking from a Java program, the document must be an instance of com.wm.app.tn.doc.BizDocEnvelope. Otherwise, the structure of *bizdoc* is defined by wm.tn.rec:BizDocEnvelope.

retriesRemaining **String** The number of retries remaining. The delivery engine passes this information into the server at every attempt of job delivery.

Output Parameters

serviceOutput **Document** The output from the delivery. For the structure of serviceOutput, see wm.tn.rec:SmtpDeliveryServiceOutput. The document contains the following keys:

- *status* **String** The outcome of the delivery: `success` or `fail`.
- *statusMessage* **String** The status message from the last attempt to deliver the document. For example, the status message might be "Mail was sent successfully."
- *transportTime* **String** Total time (in milliseconds) it took to deliver the document.
- *output* **Document** For the structure of *output*, see wm.tn.rec:SmtpOutput.

Usage Notes

The service throws an exception if the receiver identified to receive the document in `bizdoc` is a local partner. And the local partner is a host of the network.

wm.tn.transport:webService

Delivers documents through a web service. This service uses the delivery method information that is associated with the web service delivery method. The delivery method information, such as the web service connector to use, is obtained from the receiving partner's profile.

Input Parameters

serviceName	**String** The name of the delivery method that must be used to deliver the document through a web service.
bizdoc	**Document** The document you want to deliver. If invoking from a Java program, the document must be an instance of com.wm.app.tn.doc.BizDocEnvelope. Otherwise, the structure of `bizdoc` is defined by wm.tn.rec:BizDocEnvelope.
retriesRemaining	**String** (optional) The number of retries remaining. The delivery engine passes this information into the server at every attempt of job delivery.

Output Parameters

serviceOutput	**Document** The output from the delivery. For the structure of *serviceOutput*, see wm.tn.rec:WebServiceDeliveryServiceOutput. The document contains the following keys: ■ *status* **String** The outcome of the delivery: `success` or `fail`. ■ *statusMessage* **String** The status message from the last attempt to deliver the document. ■ *transportTime* **String** Total time (in milliseconds) it took to deliver the document. ■ *output* **Document** The response returned after invoking the web service. For the structure of output, see wm.tn.rec:WebServiceOutput.

21 Util Folder

Overview

The utility services (services in the wm.tn.util folder) provide services that you can use to convert java.lang.String objects to other formats.

Summary of Elements in this Folder

The following table describes the elements that are available in this folder:

Element	Description
wm.tn.util:longToSqlTimestamp	Converts a java.lang.long value to java.sql.Timestamp.
wm.tn.util:stringListToDateList	Converts a list of java.lang.String objects to a list of java.util. Date objects, using the specified date pattern.
wm.tn.util:stringListToDoubleList	Converts a list of java.lang.String objects to a list of java.util. Double objects.
wm.tn.util:stringToDate	Converts a java.lang.String object to a java.util. Date object, using the specified date pattern.
wm.tn.util:stringToDouble	Converts a java.lang.String object to a java.util. Double object.

wm.tn.util:longToSqlTimestamp

Converts a java.lang.long value to java.sql.Timestamp.

Input Parameters

longValue **String** The java.lang.long value that you want to convert to a java.sql.Timestamp object.

Output Parameters

timestamp **Object List** The converted java.sql.Timestamp value.

Usage Notes

This service uses the java.sql.Timestamp.Timestamp constructor. It constructs a *timestamp* object using a milliseconds time value. This will not throw any Exception.

wm.tn.util:stringListToDateList

Converts a list of java.lang.String objects to a list of java.util. Date objects, using the specified date pattern.

Input Parameters

list	**String List** A list of Strings you want to convert to java.util.Date objects.
pattern	**String** The date formatting pattern to use when converting the Strings.

Output Parameters

value	**Object List** The converted list of Date values.

Usage Notes

This service uses the java.text.SimpleDateFormat.parse method. If any String in the input variable *list* is unparsable using the date format supplied in the input variable *pattern*, this service will *not* throw an exception. Instead the service places a null value in the corresponding element in the output variable *value*. See the javadocs for java.text.SimpleDateFormat for a description of the *pattern* variable.

wm.tn.util:stringListToDoubleList

Converts a list of java.lang.String objects to a list of java.util. Double objects.

Input Parameters

list	**String List** A list of Strings you want to convert to java.util.Double objects.

Output Parameters

value	**Object List** The converted list of Double values.

Usage Notes

- This service uses the java.text.NumberFormat.parse method. It will throw an exception if any String in the input variable *list* is unparsable.
- If any String in *list* contains both digits and non-numeric characters, this service ignores all digits following the first non-numeric. For example, if a String contains “123x45.67”, the corresponding element in value will be 123.

wm.tn.util:stringToDate

Converts a java.lang.String object to a java.util. Date object, using the specified date pattern.

Input Parameters

string	**String** The String you want to convert to a java.util.Date object.
pattern	**String** The date formatting pattern to use when converting the String.

Output Parameters

value	**Object** The converted Date value.

Usage Notes

This service uses the java.text.SimpleDateFormat.parse method. If the value in the input variable *string* is unparsable using the date format supplied in the input variable *pattern*, this service will *not* throw an exception. Instead the service returns a null value in the output variable *value*. See the javadocs for java.text.SimpleDateFormat for a description of the *pattern* variable.

wm.tn.util:stringToDouble

Converts a java.lang.String object to a java.util. Double object.

Input Parameters

string	**String** The String you want to convert to a java.util.Double object.

Output Parameters

value	**Object** The converted Double value.

Usage Notes

- This service uses the java.text.NumberFormat.parse method. It will throw an exception if the value in the input variable *string* is unparsable.
- If the value in the input variable *string* contains both digits and non-numeric characters, this service ignores all digits following the first non-numeric. For example, if *string* contains "123x45.67", the service returns 123 in *value*.

22 Service Specifications

Overview

This section contains specifications that many of the built-in services use.

Summary of Specifications

The following table describes the specifications that many of the built-in services use:

Specification	Description
wm.tn.rec:BizDocSigningService	Use this specification for services that generate digital signatures for outbound documents.
wm.tn.rec:BizDocValidationService	Use this specification for services that structurally validate incoming documents.
wm.tn.rec:BizDocVerificationService	Use this specification for services that verify digital signatures on incoming documents.
wm.tn.rec:DeliveryServiceSignature	Use this specification for transport services for reliable delivery.
wm.tn.rec:DupCheckService	Use this specification for duplicate checking services that Trading Networks is to invoke when using the **Check for Duplicate Document** pre-processing action in a processing rule.
wm.tn.rec:ProcessingService	Use this specification for services that are invoked by the **Execute a Service** action in a processing rule.
wm.tn.rec:ReliableProcessingService	Use this specification for services that are invoked by the Execute a Service action in a processing rule.
wm.tn.rec:GatewayService	Use this specification for creating a document gateway service.
wm.tn.rec:TPAValidationService	Use this specification for services that are invoked as validation services for TPA data.

wm.tn.rec:BizDocSigningService

Use this specification for services that generate digital signatures for outbound documents.

Input Parameters

bizdoc **Document** The document to be signed. For the structure of bizdoc, see wm.tn.rec:BizDocEnvelope.

Output Parameters

errorCount	**String** The number of signing errors that the service encountered.
errors	**Document List** (optional) An array of signing-related messages, warnings, and errors that the signing service generated.

Usage Notes

- If you are invoking the signing service from a Java program, you can pass the incoming document (*bizdoc*) as an instance of com.wm.app.tn.doc.BizDocEnvelope.
- When coding a signing service, you do not need to attach the resulting message to the verified document because the document processing engine performs this action.

wm.tn.rec:BizDocValidationService

Use this specification for services that structurally validate incoming documents.

Input Parameters

bizdoc	**Document** The document to be validated. For the structure of *bizdoc*, see wm.tn.rec:BizDocEnvelope.

Output Parameters

errorCount	**String** The number of validation errors that were encountered.
errors	**Document List** (optional) An array of validation-related messages, warnings, and errors that the validation service encountered.

Usage Notes

- If you are invoking the validation service from a Java program, you can pass the incoming document (*bizdoc*) as an instance of com.wm.app.tn.doc.BizDocEnvelope.
- When coding a validation service, you do not need to attach the resulting errors to the validated document because the document processing engine does this.

wm.tn.rec:BizDocVerificationService

Use this specification for services that verify digital signatures on incoming documents.

Input Parameters

bizdoc	**Document** The document that contains the digital signature to be verified. For the structure of *bizdoc*, see wm.tn.rec:BizDocEnvelope.

Output Parameters

errorCount	**String** The number of verification errors that were encountered.
errors	**Document List** (optional) An array of verification-related messages, warnings, and errors that the verification service generated.

Usage Notes

- If you are invoking the verification service from a Java program, you can pass the incoming document (*bizdoc*) as an instance of com.wm.app.tn.doc.BizDocEnvelope.
- When coding a verification service, you do not need to attach the resulting message to the verified document because the document processing engine performs this action.

wm.tn.rec:DeliveryServiceSignature

Use this specification for transport services for reliable delivery.

Input Parameters

bizdoc	**Document** The document that is sent to the receiving partner. For the structure of *bizdoc*, see wm.tn.rec:BizDocEnvelope.
retriesRemaining	**String** The number of retries remaining. The delivery engine passes this information into the custom delivery service at every attempt of job delivery.

Output Parameters

serviceOutput	**Document** The output returned from the transport service. For the structure of *serviceOutput*, see wm.tn.rec:DeliveryServiceOutput.

Usage Notes

If you are invoking the verification service from a Java program, you can pass the incoming document (*bizdoc*) as an instance of com.wm.app.tn.doc.BizDocEnvelope. Additionally, the service will return *serviceOutput* as an instance of com.wm.data.Idata.

wm.tn.rec:DupCheckService

Use this specification for duplicate checking services that Trading Networks is to invoke when using the **Check for Duplicate Document** pre-processing action in a processing rule.

Input Parameters

bizdoc	**Document** The document that is being processed by the processing rule. For the structure of *bizdoc*, see wm.tn.rec:BizDocEnvelope.

Output Parameters

duplicate	**String** Indicates if the document is a duplicate. The service returns either `true` or `false` for *duplicate*: ■ `true` - The document represented by the BizDocEnvelope (in *bizdoc*) is a duplicate that Trading Networks has already processed. ■ `false` - The document represented by the BizDocEnvelope (in *bizdoc*) is *not* a duplicate.
message	**String** (optional) When *duplicate* is `true`, you can supply a message that you want Trading Networks to record to the activity log for the BizDocEnvelope identified in the *bizdoc* input variable. Trading Networks only saves the message if the **Save Document to Database** pre-processing action indicates that activity log information is to be saved.

Usage Notes

For a description of duplicate checking services and information about how to create them, see the chapter about processing rules in *webMethods Trading Networks Administrator's Guide*.

wm.tn.rec:ProcessingService

Use this specification for services that are invoked by the **Execute a Service** action in a processing rule. Use this specification when you specify you want to execute the service synchronous or asynchronous.

Input Parameters

bizdoc	**Document** The document that is being processed by the processing rule. For the structure of *bizdoc*, see wm.tn.rec:BizDocEnvelope.

sender	**Document** The profile summary of the sender that is identified in the document being processed (if known). For the structure of *sender*, see wm.tn.rec:ProfileSummary.
receiver	**Document** The profile summary of the receiver that is identified in the document being processed (if known). For the structure of *receiver*, see wm.tn.rec:ProfileSummary.

Output Parameters

None.

Usage Notes

If you are invoking the service from a Java program, you can pass the incoming document (*bizdoc*) as an instance of com.wm.app.tn.doc.BizDocEnvelope and *sender* and *receiver* as instances of com.wm.app.tn.profile.ProfileSummary.

wm.tn.rec:ReliableProcessingService

Use this specification for services that are invoked by the Execute a Service action in a processing rule. Use this specification when you specify you want to execute the service using a service execution task.

Input Parameters

bizdoc	**Document** The document that is being processed by the processing rule. For the structure of *bizdoc*, see wm.tn.rec:BizDocEnvelope.
sender	**Document** The profile summary of the sender that is identified in the document being processed (if known). For the structure of *sender*, see wm.tn.rec:ProfileSummary.
receiver	**Document** The profile summary of the receiver that is identified in the document being processed (if known). For the structure of *receiver*, see wm.tn.rec:ProfileSummary.

Output Parameters

serviceOutput	**Document** The output from the service. For the structure of *serviceOutput*, see wm.tn.rec:ReliableServiceOutput.

Usage Notes

If you are invoking the validation service from a Java program, you can pass the incoming document as an instance of com.wm.app.tn.doc.BizDocEnvelope and the incoming profile summaries as instances of com.wm.app.tn.profile.ProfileSummary.

wm.tn.rec:GatewayService

Use this specification for creating a document gateway service.

Input Parameters

ffdata	**Object** The flat file document as an input stream. It is sent to the receiving partner.

Output Parameters

ffdata	**Object** The flat file document as an input stream. It is sent to the receiving partner.
TN_parms	**Document** (optional) An IS document (IData object) that holds "hints" that Trading Networks uses when performing document recognition for a flat file document. See information about document gateway services in *webMethods Trading Networks Administrator's Guide* for details on providing recognition hints.

Usage Notes

Detailed information on creating a gateway service is available in the *webMethods Trading Networks Administrator's Guide*.

wm.tn.rec:TPAValidationService

Use this specification for services that are invoked as validation services for TPA data.

Input Parameters

data	**Object** An IData object containing the TPA data to be passed to the validation service for validation.

Output Parameters

output	**Document** The validation results. For the structure of *output*, see wm.tn.rec:ReliableServiceOutput.

Usage Notes

If you are writing a validation service, you can pass the incoming document (*data*) for use in validating the structure based on any existing IS document. You can use the pub.schema:validate service to validate the document and the validation result and errors in *output*.

23 IS Document Types

Summary of Elements in this Folder

The following tables list the IS document types that are available in this folder.

Document-Related to IS Document Types

These IS document types relate to business documents, TN document types, and attributes. The following table lists the IS Document types:

IS Document Type	Description
wm.tn.rec:ActivityLogEntry	An entry in the activity log.
wm.tn.rec:BizDocAttribute	A custom document attribute.
wm.tn.rec:BizDocContentPart	Content part of a document.
wm.tn.rec:BizDocContentPartCriteria	Defines the content part criteria that can be used when retrieving a BizDocEnvelope using wm.tn.doc:view.
wm.tn.rec:BizDocEnvelope	A business document.
wm.tn.rec:BizDocErrorSet	A set of errors (logged in the activity log) that are associated with a business document.
wm.tn.rec:BizDocType	A TN document type.
wm.tn.rec:BizDocTypeSummary	Summary information about a TN document type.
wm.tn.rec:ReliableServiceOutput	The pre-processing actions for a document that are specified in a TN document type or processing rule.

Profile Management IS Document Types

The following table describes the IS document types are used in the creation and maintenance of profiles for your organization and your trading partners.

IS Document Type	Description
wm.tn.rec:Address	An address for a partner on the trading network.
wm.tn.rec:Contact	A contact for a partner in the trading network.
wm.tn.rec:Corporation	The corporate component of a partner's profile on the trading network.
wm.tn.rec:Delivery	A delivery method that Trading Networks uses to deliver documents to a partner in the trading network, for example,

IS Document Type	Description
	the host name, port number, and URL Trading Networks uses to deliver a document through HTTP.
wm.tn.rec:ExternalID	An external ID for a partner on the trading network.
wm.tn.rec:Field	An extended field in a partner profile.
wm.tn.rec:FieldMetaData	Information about a profile field. wm.tn.rec:FieldMetaData is a component of wm.tn.rec:Field.
wm.tn.rec:Profile	A profile for a partner in the trading network.
wm.tn.rec:ProfileSummary	A subset of profile information for a partner in the trading network.

Security-Related IS Document Types

The following table describes the IS document types relate to security certificate handling:

IS Document Type	Description
wm.tn.rec:SmtpDeliveryServiceOutput	The output from the SMTP (e-mail) delivery services, for example, wm.tn.transport:primaryFtp.

Task Management Service IS Document Types

The following table describes the IS document types are used in the services related to tasks that use reliable delivery (delivery tasks) and reliable execution (service execution tasks):

IS Document Type	Description
wm.tn.rec:DeliveryService	A delivery service.
wm.tn.rec:DeliveryServiceOutput	The output from a delivery service.
wm.tn.rec:ReliableServiceOutput	The output from a service that is being executed by a service execution task.
wm.tn.rec:Task	A task.
wm.tn.rec:TaskDbUpdate	If the Trading Networks property, `tn.task.dbupdate.retryEnabled` is set to `true`, Trading Networks publishes a document of this type when it attempts to retry updating its database with information for a task.
wm.tn.rec:TaskFailure	If the task failure notification feature is enabled, Trading Networks publishes a document of this type.

TPA IS Document Types

The following table describes the IS document type relates to trading partner agreements (TPAs).

IS Document Type	Description
wm.tn.rec:tpa	A trading partner agreement (TPA).
wm.tn.rec:tpaError	An IData error object.
wm.tn.rec:TPAValidateServiceOutput	The record reference to which validation services must adhere in order to pass validation output results.

Delivery Service IS Document Types

The following table describes the IS document types can be used with the transport services for ActiveTransfer, HTTP, HTTPS, FTP, FTPS, SMTP, and Web service.

Is Document Type	Description
wm.tn.rec:activeTransferOutput	If a task uses the ActiveTransfer delivery service to deliver documents, use this IS document type to map the output from the delivery service.
wm.tn.rec:FtpDeliveryServiceOutput	The output from the FTP or FTPS delivery service, for example, wm.tn.transport:Ftp.
wm.tn.rec:FtpOutput	If a task uses a delivery service to deliver the document, use this IS document type to map the output from the delivery service.
wm.tn.rec:HttpDeliveryServiceOutput	The output from the HTTP or HTTPS delivery service, for example, wm.tn.transport:Http.
wm.tn.rec:HttpOutput	If a task uses a delivery service to deliver the document, use this IS document type to map the output from the delivery service.
wm.tn.rec:SmtpDeliveryServiceOutput	The output from the SMTP (e-mail) delivery services, for example, wm.tn.transport:primaryFtp.
wm.tn.rec:SmtpOutput	If a task uses a delivery service to deliver the document, use this IS document type to map the output from the delivery service.
wm.tn.rec:SftpOutput	If a task uses a delivery service to deliver the document, use this IS document type to map the output from the delivery service.

Is Document Type	Description
wm.tn.rec:WebServiceDeliveryServiceOutput	The output from the delivery service that Trading Networks provides for its web service delivery methods.
wm.tn.rec:WebServiceOutput	If a task uses a wm.tn.transport:webService delivery service to deliver the document, use this IS document type to map the output from the delivery service.

Query-Related IS Document Types

The following table describes the IS document types relate to the services that are used to query the Trading Networks database for information about documents and activity log entries.

IS Document Type	Description
wm.tn.rec:queryField	The document structure of the *fields* variable in the wm.tn.rec:queryInput IS document type.
wm.tn.rec:queryInput	The input for a query.
wm.tn.rec:queryOutput	Contains the query execution details.
wm.tn.rec:svcResponse	Contains the response from a service.

wm.tn.rec:ActivityLogEntry

An entry in the activity log.

Variables

EntryTimestamp **String** The time the activity log entry was created.

EntryType **String** Type of the entry. Valid values for *EntryType* are:

- `0` - For error
- `1` - For warning
- `2` - For message

EntryClass **String** The category (or activity class) for the entry. The value can be any string from 1-20 characters. Trading Networks uses the following activity classes:

- Delivery
- Persistence

- Recognition
- Processing
- Validation
- Verification
- General

For a description of the activity classes that Trading Networks uses, see information about using the activity log in *webMethods Trading Networks User's Guide*.

BriefMessage	**String** The brief message for the activity log entry. The value can be any string from 1-80 characters.
FullMessage	**String** (optional) A more detailed message for the activity log entry. The value can be any string from 1-1024 characters.
RelatedDocID	**String** (optional) The internal document ID of the document related to this activity log entry.
RelatedPartnerID	**String** (optional) The partner ID for the partner related to this activity log entry.
RelatedConversationID	**String** (optional) The Conversation ID related to this activity log entry.
RelatedStepID	**String** (optional) The Step ID of the conversation related to this activity log entry.
B2Buser	**String** (optional) The user name of the current user when this activity log entry was created.

wm.tn.rec:Address

An address for a partner on the trading network. In the Trading Networks database, an address can be associated with a corporation or a contact.

Variables

AddressID	**String** An internal identifier for an address. For the format of *AddressID*, see wm.tn.rec:Field.
PartnerID	**String** (optional) An internal identifier for a partner in the trading network. If the address is associated with a corporation, this field should have a value and *ContactID* should not. For the format of *PartnerID*, see wm.tn.rec:Field.

ContactID	**String** (optional) An internal identifier for a partner's contact. If the address is associated with a contact, this field should have a value and *PartnerID* should not. For the format of *PartnerID*, see wm.tn.rec:Field.
AddressLine1	**String** The first line of the address. For the format of *AddressLine1*, see wm.tn.rec:Field.
AddressLine2	**String** The second line of the address. For the format of *AddressLine2*, see wm.tn.rec:Field.
AddressLine3	**String** The third line of the address. For the format of *AddressLine3*, see wm.tn.rec:Field.
City	**String** The city in the address. For the format of *City*, see wm.tn.rec:Field.
State_Province	**String** The state or province for the address. For the format of *State_Province*, see wm.tn.rec:Field.
Zip_PostalCode	**String** The ZIP code or postal code for the address. For the format of *Zip_PostalCode*, see wm.tn.rec:Field.
Country	**String** The country for the address. For the format of *Country*, see wm.tn.rec:Field.
SequenceNumber	**Object** The sequence number of the address. Trading Networks uses *SequenceNumber* to display corporate addresses in the proper order in the user interface. For the format of *SequenceNumber* see wm.tn.rec:Field. The data type of the variable *Value* in wm.tn.rec:Field is java.lang.Short.

wm.tn.rec:BizDocAttribute

A custom document attribute. These are usually instances of com.wm.app.tn.doc.BizDocAttribute.

Variables

AttributeID	**String** An internal unique identifier for this document attribute.
AttributeName	**String** A name for this document attribute. The value can be a string from 1-64 characters.
AttributeDescription	**String** A description of this document attribute. The value can be a string from 1-256 characters.
AttributeType	**String** The data type of this document attribute. The value can be one of the following: ■ STRING ■ NUMBER

- DATETIME
- STRING LIST
- NUMBER LIST
- DATETIME LIST

Deleted? **String** Whether this document attribute is deleted. Valid values are:

- `true` The attribute is marked as deleted.
- `false` The attribute is *not* marked as deleted.

Persist? **String** Whether or not the attribute is to be saved. Valid values are:

- `true` The attribute is to be saved.
- `false` The attribute is *not* to be saved.

LastModified **String** The timestamp when this document attribute was last saved.

wm.tn.rec:BizDocContentPart

Content part of a document. These are usually instances of com.wm.app.tn.doc.BizDocContentPart.

Variables

PartName **String** A name for this content part. The value can be a string from 1-100 characters.

MimeType **String** Mime type of this part. The value can be a string from 1-100 characters.

Length **Integer** Size of this part in bytes.

Bytes **Object** Content of this part. This should be a byte[].

StorageType **String** The name of the storage system where the content of the document is stored. The value is a string from 1-100 characters and is used when Trading Networks considers the content part to be large. The value is "tspace" when the content of this content part is stored in the webMethods temporary storage system called, TSpace storage system.

StorageRef **Object** When the content part is considered large, specifies the reference pointer to the storage system where content of the part is stored. This should be of type java.lang.Object. When the content is stored in the TSpace system, this field contains an instance of com.wm.util.tspace.Reservation object.

LargePart?	**String** Indicates whether Trading Networks considers this content part to be large. Use this value to determine whether this content part requires large document handling. Valid values are: - `true` - The content part is large; that is, its content is stored in the storage system indicated by *StorageType* and *StorageRef* and is *not* stored in *Bytes*. - `false` - The content part is not large; that is, its content is stored in *Bytes*.

Usage Notes

For more information about large document handling, including how Trading Networks determines whether a document is large, see *webMethods Trading Networks Administrator's Guide*.

wm.tn.rec:BizDocContentPartCriteria

Defines the content part criteria that can be used when retrieving a BizDocEnvelope using wm.tn.doc:view.

Using this IS document type, you can control what content parts you want to retrieve with the document. You can specify a list of part names to be included and a list of part names to be excluded from the retrieved envelope.

Variables

includeParts	**String List** (optional) String array consisting of names of the content parts that should be included in the retrieved envelope.
ExcludeParts	**String List** (optional) String array consisting of names of the content parts that should be excluded from the retrieved envelope.

wm.tn.rec:BizDocEnvelope

A business document.

These are usually instances of com.wm.app.tn.doc.BizDocEnvelope.

Variables

InternalID	**String** The internal document ID. This is an internal unique identifier for this document.
DocType	**Document** The TN document type of this document. For the structure of *DocType*, see wm.tn.rec:BizDocType.

DocTimestamp	**String** The timestamp when Trading Networks received this document.
LastModified	**String** The timestamp when this document was last modified.
SenderID	**String** The internal partner ID for the sender of this document. That is, the value to which Trading Networks transformed the value of the *TN_parms/SenderID* variable that the document gateway service extracted from the flat file.
ReceiverID	**String** The internal partner ID for the receiver of this document. That is, the value to which Trading Networks transformed the value of the *TN_parms/ReceiverID* variable that the document gateway service extracted from the flat file.
DocumentID	**String** (optional) An external (user-defined) identifier for this document. This is the document ID identified within a document.
GroupID	**String** (optional) An external (user-defined) identifier for this document's group.
ConversationID	**String** (optional) An external (user-defined) identifier for this document's conversation.
SystemStatus	**String** The processing status of the document. The value will be one of the following: ■ NEW ■ DONE ■ DONE W/ ERRORS ■ POLLABLE ■ ACCEPTED ■ ACCEPTED W/ ERRORS For a description of the processing statuses, see *webMethods Trading Networks Administrator's Guide*.
UserStatus	**String** (optional) The user-defined status of the document. The value can be any string from 1-20 characters.
Attributes	**Document** (optional) The custom attributes for this document, keyed by attribute ID.
Signature	**Object** The digital signature for this document.
SignatureBody	**Object** The portions of the document that was used to create the digital signature.
ContentParts	**Document List** The content parts in this document. Each content part is formatted as a wm.tn.rec:BizDocContentPart.

Content **Object** Deprecated. The original content of the document is in the list of *ContentParts*.

Persisted? **String** Whether this document has been saved to the database. Valid values are:

- `true` - The document was saved to the database.
- `false` - The document was *not* saved to the database.

Errors **Document** A set of activity log entries associated with this document, sorted by entry class. For the structure of *Errors*, see wm.tn.rec:BizDocErrorSet.

Relationships **Document List** List of the documents related to this one.

LargeDocument? **String** Whether Trading Networks considers this to be a large document. If *LargeDocument?* is `true`, some of parts of this document might require large document handling. Valid values are:

- `true` - The document is large; that is, at least one of its content parts is considered large.
- `false` - The document is *not* large; that is, none of its content parts are large.

OriginalSenderID **String** The external partner ID for the sender of this document. That is, the original value of the *TN_parms/SenderID* variable that the document gateway service extracted from the flat file.

OriginalReceiverID **String** The external partner ID for the receiver of this document. That is, the original value of the *TN_parms/ReceiverID* variable that the document gateway service extracted from the flat file.

MonitoringEnabled **String** Whether the Trading Networks document is enabled for BAM. If so, Trading Networks sends the document data to webMethods Optimize for B2B for monitoring. For more information about BAM on Trading Networks data, see *webMethods Optimize for B2B User's Guide*.

Valid values are:

- `true` - The document is enabled for BAM.
- `false` - The document is *not* enabled for BAM. Default.

MonitoringDataItems **Hashmap** The attributes for monitoring and their values that were extracted from the document. This data is sent as an event to Optimize for B2B, after the document routing is complete.

SetMonitoringAction **String** Whether Trading Networks must consider document routing to be complete after the **Execute a Service** action completion or after the **Deliver Document By** action completion. This is applicable only when both **Execute a Service** and **Deliver Document By** actions are specified

	in the processing rule. Trading Networks sends the monitoring data to Optimize for B2B after completion of document routing. Valid values are: ■ `Service execution` - Trading Networks considers the end of document routing to be when the **Execute a Service** action is complete. ■ `Deliver` - Trading Networks considers the end of document routing to be when the **Deliver Document By** action is complete.
RepeatNum	**String** The number of times the document has been reprocessed and resubmitted.
RoutingType	**String** Whether the document was reprocessed or resubmitted.
VERSION_NO	**String** Version of Trading Networks.
ReceiveSvc	**String** For resubmitting the document. This is extracted from *TN_parms/$receiveSvc* from the pipeline.
Comments	**String** Comments associated with transactions.
Duplicate	**String** Whether the document is duplicate or not. Valid values are `true` and `false`.

Usage Notes

For more information about large document handling, including how Trading Networks determines whether a document is large, see *webMethods Trading Networks Administrator's Guide*.

wm.tn.rec:BizDocErrorSet

A set of errors (logged in the activity log) that are associated with a business document. These are usually instances of com.wm.app.tn.doc.BizDocErrorSet.

Variables

The exact keys for this variable vary from document to document. Each key will be an entry class (also known as an activity class), for example, *Validation*, *Verification*, *Processing*, or *General*.	**Document List** A list of activity log entries of this class belonging to this error set. For the structure of each activity log entry in the document list, see wm.tn.rec:ActivityLogEntry.

wm.tn.rec:BizDocType

A TN document type. These are usually instances of com.wm.app.tn.doc.BizDocType.

Variables

TypeID	**String** An internal unique identifier for this TN document type
TypeName	**String** A name for this TN document type. The value can be a string from 1-64 characters.
TypeDescription	**String** A description of this TN document type. The value can be a string from 1-256 characters.
Deleted?	**String** Whether this TN document type is deleted. Valid values are: ■ `true` The TN document type is marked as deleted. ■ `false` The TN document type is *not* marked as deleted.
LastModified	**String** The timestamp when this TN document type was last saved.
PreRoutingFlags	**Document** Pre-processing actions defined in this TN document type. For the structure of *PreRoutingFlags*, see wm.tn.rec:ReliableServiceOutput.

wm.tn.rec:BizDocTypeSummary

Summary information about a TN document type.

Variables

TypeID	**String** An internal unique identifier for this TN document type.
TypeName	**String** A name for this TN document type. The value can be a string from 1-64 characters.
TypeDescription	**String** A description of this TN document type. The value can be a string from 1-256 characters.
Deleted?	**String** Whether this TN document type is deleted. Valid values are: ■ `true` - The TN document type is marked as deleted. ■ `false` - The TN document type is *not* marked as deleted.
LastModified	**String** The timestamp when this TN document type was last saved.

wm.tn.rec:Contact

A contact for a partner in the trading network.

At least one contact must be defined for a partner's profile to be activated.

Variables

PartnerID	**String** An internal identifier of a partner in the trading network.
ContactID	**String** An internal identifier of the contact.
GivenName	**String** The given name of the contact.
Surname	**String** The surname (for example, last name) of the contact.
SequenceNumber	**Object** The sequence number of the contact. Trading Networks uses this to display the contacts in the proper order in the user interface. The data type of the variable is java.lang.Short.
Type	**Object** The type of contact. The data type of the variable is java.lang.Short. Its value must be one of the Contact Types known to your trading Network. The following are the built-in Contact Types, but you can create additional ones with the wm.tn.dictionary:addContactType service. ■ 1 - Technical contact ■ 2 - Administrative contact
Role	**String** The contact's role in the organization. Its value can be any string from 1-64 characters.
EmailAddress	**String** The contact's e-mail address.
TelNumber	**String** The contact's telephone number.
TelExtension	**String** The contact's telephone extension.
FaxNumber	**String** The contact's facsimile number.
PagerNumber	**String** The contact's pager number.
Address	**Document** The contact's address. For the structure of *Address*, see wm.tn.rec:Address.

wm.tn.rec:Corporation

The corporate component of a partner's profile on the trading network.

Variables

PartnerID	**String** An internal identifier of a partner in the trading network.
CorporationName	**String** The name of the corporation.
OrgUnitName	**String** The name of the organizational unit within the corporation.

Status **String** The status of the partner in your Integration Server. Its value can be either `Active` or `Inactive`.

Type **String** The type of software the partner uses to connect to your network. Its value can be one of the following:

- TNServer (webMethods Trading Networks)
- TNPartner (webMethods for Partners)
- Browser (web browser)
- Other

Self **Object** Indicates if the partner profile represents the local partner (the host of the trading network). The data type of the variable is com.wm.data.MBoolean.

Deleted **Object** Indicates if the partner has been marked as deleted. The data type of the variable is com.wm.data.MBoolean.

TNVersion **String** The Trading Networks version used by the partner.

RemoteStatus **String** Not currently used.

Certificate **String** [deprecated] null. Use the new service wm.tn.security:getCertificateData for certificate.

CACertificate **String** [deprecated] null. Use the new service wm.tn.security:getCertificateData for CA certificates.

PrivateKey **String** [deprecated] null. Use the new service wm.tn.security:getCertificateData for private key.

PreferredProtocol **String** The delivery protocol that the partner prefers you to use when sending documents to it. Valid values are:

- `ftp1` - Primary FTP
- `ftp2` - Secondary FTP
- `http1` - Primary HTTP
- `http2` - Secondary HTTP
- `https1` - Primary HTTPS
- `https2` - Secondary HTTPS
- `smtp1` - Primary Email
- `smtp2` - Secondary Email
- <null> Polling

- *<name of the delivery method that you created>* - The delivery method that you created using the delivery protocols that Trading Networks provides.

PollingFrequency **Object** How often (in minutes) the partner will poll for documents that are queued on your Integration Server. The data type of the variable is java.lang.Float.

DeliveryMaxRetries **Object** If the delivery of a document to the partner fails, how many times to retry to deliver the document. The data type of the variable is java.lang.Short. (This setting is only used when reliable delivery is in use. Trading Networks only uses reliable delivery for a document if the document is saved to the database.)

DeliveryRetryWait **Object** The number of milliseconds you want Trading Networks to wait before making its first attempt to redeliver the document (if the original attempt to deliver the document fails). The data type of the variable is java.lang.Integer. (This setting is only used when reliable delivery is in use. Trading Networks only uses reliable delivery for a document if the document is saved to the database.)

PollingProtocol **String** The delivery protocol the partner uses to poll for documents on your Integration Server.

RetryFactor **Object** The factor you want Trading Networks to use when calculating how long to wait before making the second and subsequent attempts to redeliver the document. Trading Networks calculates the time by multiplying the last wait time by *RetryFactor*. Specify a whole number grater than zero for *retryFactor*. The data type of the variable is java.lang.Integer.

CompanyLogo **Object** The corporate logo in GIF or JPEG format. The data type of the variable is byte[].

Address **Document** The corporate address. For the structure of *Address*, see wm.tn.rec:Address.

PreferredLocale **String** The locale for the profile.

RoutingOff **String** Whether document delivery is suspended for the partner. Valid values are:

- `true` - Delivery is suspended for the partner.
- `false` - Delivery is *not* suspended for the partner.

wm.tn.rec:Delivery

A delivery method that Trading Networks uses to deliver documents to a partner in the trading network, for example, the host name, port number, and URL Trading Networks uses to deliver a document through HTTP.

Variables

PartnerID **String** An internal identifier of a partner in the trading network.

Protocol **String** The communications protocol to use. Its value can be one of the following:

- `ftp`
- `ftps`
- `http`
- `https`
- `smtp`
- *<name of the delivery method that you create>*

The values `ftp`, `ftps`, `http`, `https`, and `smtp` are valid only for Primary and Secondary built-in immediate delivery methods.

PrimaryAddr **Object** Whether the delivery method is associated with the primary or secondary address, for example, Primary HTTP or Secondary HTTP. The data type of the variable is com.wm.data.MBoolean; its value can be one of the following:

- `1` - Primary address
- `0` - Not a Primary address (Secondary)

> **Note:**
> These values are valid only for the default registered services such as wm.tn.transport:primaryHttp.

Host **String** The host name (for example "yourcompany.com"). If *Protocol* is `smtp` (e-mail), *Host* is ignored.

Port **String** The port number of the communications port. If *Protocol* is `smtp` (e-mail), *Port* is ignored.

Location **String** The location to which to deliver the document. For the following protocol values, the location is:

- `ftp` or `ftps` - The directory to which to put the document.
- `http` or `https` - The URL to which to send the document.
- `smtp` - The email address to which to send the document.

Username **String** The default user name for a delivery method.

Password **String** The default password for a delivery method.

CustomData	**Object** The user-defined custom data for the delivery method. *CustomData* holds additional data for custom delivery mechanisms. The data type of the variable is byte[].
DestinationID	**String** An internal identifier of the delivery method.
B2BService	The name of the custom delivery service.
B2BInterface	The fully qualified name of the custom delivery service folder.

wm.tn.rec:DeliveryService

A delivery service.

Variables

Name	**String** The name used to register this service.
Interface	**String** The fully-qualified name of the folder for the delivery service.
Service	**String** The service name for the delivery service.
Location	**String** The location to which to deliver the document.
Local	**Object** Whether or not this service is a local or remote service. The data type of *Local* is com.wm.data.MBoolean.

wm.tn.rec:DeliveryServiceOutput

The output from a delivery service.

Variables

status	**String** The status the delivery service returned. The value of *status* is either `success` or `fail`.
statusMessage	**String** The delivery-specific message that the delivery service returned along with the *status*.
transportTime	**String** The total time (in milliseconds) that the delivery service used to deliver the document.
output	**Document** Return information, if applicable; otherwise, null. The format is specific to the delivery service used. See the documentation for the specific delivery service for the format.

wm.tn.rec:ExternalID

An external ID for a partner on the trading network. When exchanging documents, partners typically identify themselves using some well-known ID scheme, such as a D-U-N-S number.

Variables

ExternalID	**String** The external ID.
IDType	**Object** The type of ID. The data type of the variable is java.lang.Integer; to determine the value, use the wm.tn.dictionary:getIDTypes.
InternalID	**String** The internal identifier (partner ID) of the partner whose external ID this is.
SequenceNumber	**Object** The sequence number of the external ID. Trading Networks uses *SequenceNumber* to display the external IDs in the proper order in the user interface. The data type of the variable is java.lang.Short.
PartnerIDID	**String** An internal identifier of the partner's external ID.

wm.tn.rec:Field

An extended field in a partner profile. You create extended profile fields using My webMethods.

Variables

Value	**String or Object** The value of the field. The type is defined by the value of *Datatype* in *MetaData*.
PartnerID	**String** The partner ID for the partner whose field this is.
MetaData	**Document** See wm.tn.rec:FieldMetaData.

wm.tn.rec:FieldMetaData

Information about a profile field. wm.tn.rec:FieldMetaData is a component of wm.tn.rec:Field. All profile fields have metadata. You define the metadata for extended fields when you define profile fields using My webMethods.

Variables

GroupID	**Object** The field group to which the field belongs. *GroupID* is a java.lang.Integer. Specify one of the following values to indicate one of the built-in groups that Trading Networks provides. Valid values are:

- 1 - Corporate group
- 2 - Contact group
- 3 - Delivery group
- 4 - Custom group
- 5 - ID group
- 6 - Address group

You can extend the set of built-in groups by using wm.tn.dictionary:addFieldGroup.

MaxLength **Object** The maximum length for the field's value. A value of -1 indicates there is no maximum length. *MaxLength* is a java.lang.Integer.

Extended? **Object** Whether the field is an extended field. *Extended?* is a com.wm.data.MBoolean. You do not need to set the value of this field. All fields that you create are extended fields. Any value in this field is ignored.

Required? **Object** Whether the field is required. *Required?* is a com.wm.data.MBoolean. Setting this to `true` causes Trading Networks to fail validation of the field if the field's value is null.

Registration? **Object** Not currently used.

Deleted? **Object** Whether the field is deleted. *Deleted?* is a com.wm.data.MBoolean. You cannot use My webMethods or any services to physically remove an extended field from the database. You can set this to `true` to mark a field deleted. When a field is marked as deleted, Trading Networks no longer displays the field in any profiles. This value applies only to extended fields.

ValidValues **String** (optional) A concatenation of all valid value strings for this profile extended field delimited by "|~~|". This value applies only to extended fields.

DefaultValue **String** (optional) Default value for profile extended field. This value applies only to extended fields.

Datatype **String** Describes the data type of the value for the field. For extended fields, the value is one of the following:

- String
- Binary

Changing this value for standard fields has no effect.

Name **String** (optional) The name of the field. This value applies only to extended fields.

Table	**String** (optional) The name of the table in the Trading Networks database in which this field resides. This value applies only to standard fields.
Column	**String** (optional) The name of the column in the Trading Networks database in which this field resides. This value applies only to standard fields.
Description	**String** (optional) A description of the meaning and/or purpose of the field.
ProfileFieldID	**String** An internal identifier of the profile field.
Displayable?	**Object** Indicates whether Trading Networks should display the field in My webMethods. This value applies only to standard fields.

wm.tn.rec:activeTransferOutput

The output from the ActiveTransfer delivery service, for example, "wm.tn.transport:activeTransfer" on page 289.

Variables

status	**String** The status that the delivery service returns. The value of the *status* is either `success` or `fail`.
statusMessage	**String** The delivery-specific message that the delivery service returns along with the *status*.
transportTime	**String** The total time (in milliseconds) that the delivery service uses to deliver the document.
vfsPath	**String** The VFS location that the document is delivered.
output	**Document** The response from the wm.mft.external.tn:deliverDocument delivery service. For the structure of the *output*, see *webMethods ActiveTransfer Built-In Services Reference*.

wm.tn.rec:FtpDeliveryServiceOutput

The output from the FTP or FTPS delivery service, for example, wm.tn.transport:Ftp.

Variables

status	**String** The status the delivery service returned. The value of the *status* is either `success` or `fail`.

statusMessage	**String** The delivery-specific message that the delivery service returned along with the *status*.
transportTime	**String** The total time (in milliseconds) that the delivery service used to deliver the document.
output	**Document** Return information from the delivery service. For the structure of *output*, wm.tn.rec:FtpOutput.

wm.tn.rec:FtpOutput

If you know that a task uses one of the following delivery services to deliver the document, you can use this Integration Server document type to map the output from the delivery service.

- wm.tn.transport:Ftp
- wm.tn.transport:Ftps
- wm.tn.transport:primaryFtp
- wm.tn.transport:primaryFtps
- wm.tn.transport:secondaryFtp
- wm.tn.transport:secondaryFtps

Variables

returncode	**String** The standard FTP protocol return code.
returnmsg	**String** The standard FTP protocol return message.
logmsg	**String** The FTP log messages.

wm.tn.rec:SftpDeliveryServiceOutput

The output from the SFTP delivery service, for example, wm.tn.transport:Sftp.

Variables

status	**String** The status the delivery service returned. The value of the *status* is either `success` or `fail`.
statusMessage	**String** The delivery-specific message that the delivery service returned along with the *status*.
transportTime	**String** The total time (in milliseconds) that the delivery service used to deliver the document.

output	**Document** Return information from the delivery service. For the structure of *output*, wm.tn.rec:SftpOutput.

wm.tn.rec:SftpOutput

If you know that a task uses one of the following delivery services to deliver the document, you can use this Integration Server document type to map the output from the delivery service.

- wm.tn.transport:Sftp
- wm.tn.transport:primarySftp
- wm.tn.transport:secondarySftp

Variables

returncode	**String** The standard SFTP protocol return code.
returnmsg	**String** The standard SFTP protocol return message.
logmsg	**String** The SFTP log messages.

wm.tn.rec:HttpDeliveryServiceOutput

The output from the HTTP or HTTPS delivery service, for example, wm.tn.transport:Http.

Variables

status	**String** The status the delivery service returned. The value of *status* is either `success` or `fail`.
statusMessage	**String** The delivery-specific message that the delivery service returned along with the status.
transportTime	**String** The total time (in milliseconds) that the delivery service used to deliver the document.
output	**Document** Return information from the delivery service. For the structure of *output*, see wm.tn.rec:HttpOutput.

wm.tn.rec:HttpOutput

If you know that a task uses one of the following delivery services to deliver the document, you can use this Integration Server document type to map the output from the delivery service.

- wm.tn.transport:Http

- wm.tn.transport:Https
- wm.tn.transport:primaryHttp
- wm.tn.transport:primaryHttps
- wm.tn.transport:secondaryHttp
- wm.tn.transport:secondaryHttps

Variables

encodedURL	**String** The final URL of the delivered document.
header	**Object** An object that represents the HTTP response header.
lines	**Document** The response header. Each entry in *lines* represents a field (line) of the response header. The entry's name is the field name and the entry's value is the value of the field.
status	**String** The status code of the response.
statusMessage	**String** The status message of the response.
body	**Document** A byte array that contains the HTTP response data. The document contains the following variables: - *stream* **Object** A stream that contains the data from the HTTP response. *stream* is returned if Trading Networks requested the response to be returned as a stream. - *bytes* **byte[]** A byte array that contains the data from the HTTP response. *bytes* is returned if Trading Networks requested the response to be returned as bytes.

wm.tn.rec:PreProcessingFlags

The pre-processing actions for a document that are specified in a Trading Networks document type or processing rule.

These are typically instances of com.wm.app.tn.route.PreRoutingFlags.

Variables

verify?	**String** Whether documents should be verified. Valid values are: - `yes` Documents should be verified. - `no` Documents should *not* be verified.

- `don't care` Use the setting specified in the TN document type for the document being processed.

validate? **String** Whether documents should be validated. Valid values are:

- `yes` Documents should be validated.
- `no` Documents should *not* be validated.
- `don't care` Use the setting specified in the TN document type for the document being processed.

persist? **String** Whether documents should be saved to the database. Valid values are:

- `yes` Documents should be saved to the database.
- `no` Documents should *not* be saved to the database.
- `only if unique` Documents are saved *only* if they are unique.
- `don't care` Use the setting specified in the TN document type for the document being processed.

persistOption? **String** The data that Trading Networks is to save for the document. Valid values are:

- `content, attributes and activity log` - Trading Networks saves all data associated with the document; that is, Trading Networks saves the document content, the values it extracted for the custom attributes, and the activity log entries that relate to the document.
- `content only` - Trading Networks saves only the document content. Trading Networks does *not* save the values of any extracted custom attributes or the related activity log entries.
- `attributes only` - Trading Networks saves only the values it extracts for the custom attributes. Trading Networks does *not* save the document content or the related activity log entries.
- `activity log only` - Trading Networks saves only the activity log entries that are related to the document. Trading Networks does *not* save the document content or the values of any extracted custom attributes.
- `content and attributes` - Trading Networks saves the document content and the values of all extracted custom attributes. Trading Networks does *not* save the related activity log entries.
- `attributes and activity log` - Trading Networks saves the values of all extracted custom attributes and the activity log entries that relate to the document. Trading Networks does *not* save the document content.

- `don't care` - Trading Networks defers to the settings specified in the TN document type to determine the data to save for the document.
- `persist none` Trading Networks saves no data associated with the document.

unique

String Whether documents should checked to determine if they are unique.

Document uniqueness criteria. Valid values: “don't care”, “Native ID only”, “Native ID and sender.”

Valid values are:

- `DocumentID only` - The document is checked for uniqueness. To determine if the document is unique, Trading Networks determines whether another document with the same document ID exists in the database. (The document ID is a user-defined, external identifier for the document.)
- `DocumentID and sender` - The document is checked for uniqueness. To determine if the document is unique, Trading Networks determines whether another document with the same document ID and sender exists in the database.
- `DocumentID, sender and receiver` - The document is checked for uniqueness. To determine if the document is unique, Trading Networks determines whether another document with the same document ID, sender, and receiver exists in the database.
- `DocumentID, sender and document type` - The document is checked for uniqueness. To determine if the document is unique, Trading Networks determines whether another document with the same document ID, sender, and TN document type exists in the database.
- `don't care` - Use the setting specified in the TN document type for the document being processed.

wm.tn.rec:Profile

A profile for a partner in the trading network.

A partner's profile can be obtained by invoking the wm.tn.profile:getProfile. The profile contains all the standard fields. To obtain extended fields, invoke the wm.tn.profile:getExtendedFields.

Variables

Corporate

Document The corporate data. For the structure of *Corporate*, see wm.tn.rec:Corporation.

Contact	**Document List** A list of contacts. For the structure of each a contact in the list, see wm.tn.rec:Contact.
Delivery	**Document List** A list of delivery methods. For the structure of the delivery method information in the list, see wm.tn.rec:Delivery.
ID	**Document List** A list of external IDs. For the structure of each external ID in the list, see wm.tn.rec:ExternalID.
ProfileGroups	**String List** The names of the partner groups that the partner is a member of.
users	**String List** The use names of the My webMethods and Integration Server user accounts to which this profile is mapped.

wm.tn.rec:ProfileSummary

A subset of profile information for a partner in the trading network.

Variables

ProfileID	**String** The partner ID for the partner in the network.
CorporationName	**String** The name of the corporation. The value can be a string from 1-64 characters.
OrgUnit	**String** The name of the organizational unit within the corporation. The value can be any string from 1-64 characters.
Type	**String** The type of software the partner uses to connect to your network. The value can be one of the following: ■ TNServer (webMethods Trading Networks) ■ TNPartner (webMethods for Partners) ■ Browser (web browser) ■ Other
Self?	**Object** Indicates if the partner profile represents the Enterprise profile (the host of the trading network). *Self?* is a com.wm.data.Boolean and can have one of the following values: ■ true ■ false
Status	**String** The status of the partner in your Integration Server. The value can be either `Active` or `Inactive`.

RemoteStatus **String** Not currently used.

PreferredProtocol **String** The delivery protocol that the partner prefers for documents. Valid values are:

- `ftp1` - Primary FTP
- `ftp2` - Secondary FTP
- `http1` - Primary HTTP
- `http2` - Secondary HTTP
- `https1` - Primary HTTPS
- `https2` - Secondary HTTPS
- `smtp1` - Primary Email
- `smtp2` - Secondary Email
- <null> Polling
- *<name of the delivery method that you created>* - The delivery method that you created using the delivery protocols that Trading Networks provides.

PollingProtocol **String** The delivery protocol the partner uses to poll for documents on your Integration Server. Valid values are:

- `http1` - Primary HTTP
- `http2` - Secondary HTTP
- `https1` - Primary HTTPS
- `https2` - Secondary HTTPS

TNVersion **String** The version of Trading Networks that the partner uses. The data type of the variable *Value* in wm.tn.rec:Field is java.lang.String.

Deleted? **Object** Whether the partner has been marked as deleted. *Deleted?* is a com.wm.data.MBoolean.

TimeToWait **Object** The number of milliseconds you want Trading Networks to wait before making its first attempt to redeliver a document (if the original attempt to deliver the document fails). The data type of the variable *Value* in wm.tn.rec:Field is java.lang.Short.

(Trading Networks uses *TimeToWait* along with *RetryFactor* to calculate how long to wait for subsequent retry attempts.)

RetryLimit **Object** If the delivery of a document to the partner fails, how many times Trading Networks is to retry to deliver the document. The data type of the variable *Value* in wm.tn.rec:Field is java.lang.Short.

RetryFactor **Object** The factor Trading Networks should use when calculating how long to wait before making the second and subsequent attempts to redeliver the document. Trading Networks calculates the time to wait by multiplying the last wait time by *retryFactor*. Specify a whole number greater than zero for *RetryFactor*. The data type of the variable *Value* in wm.tn.rec:Field is java.lang.Integer.

ProfileGroups **String List** The names of the partner groups of which the partner is a member.

RoutingStatusOff? **String** Whether document delivery is suspended for the partner. Valid values are:

- `true` - Delivery is suspended for the partner.
- `false` - Delivery is *not* suspended for the partner. This is the default.

wm.tn.rec:queryField

The document structure of the *fields* variable in the wm.tn.rec:queryInput IS document type.

Variables

fieldName **String** Name of the field for which you specify the filter criteria. The column names can be:

- Any custom attribute ID of a Document Type.
- Any of the following Strings based on the query type.

For the list of valid values, see "List of Column names" on page 352.

isOrQuery **String** Whether the join condition for the filter criteria in the *criteria* variable is `AND` or `OR`. Valid values for the JOIN condition:

- `true` - OR
- `false` - AND (Default)

criteria **Document List** The filter criteria. Each field can have more than one filter criteria. For example, `UserStatus IS NULL AND UserStatus = IGNORED`,

where `UserStatus` is the field name, `IS NULL` and '=' are the operators, `AND` is the join condition, and `IGNORED` is the value.

Each document in the document list contains the following variables:

- *operator* **String** The operator for the filter criteria. The valid values depend on the field type of the *fieldName* variable. If you specify a sub-query (*dynamicCriteria*), specify the operator value as `Dynamic`.

For the list of valid values based on the field types, see "Valid Values for the criteria field variable" on page 354.

- *value* **String** The value for the filter criteria.

  For example, for the filter criteria, '`UserStatus = IGNORED`', `UserStatus` is the field name, '=' is the operator, and `IGNORED` is the value.

  If you specify a sub-query (*dynamicCriteria*), then to specify the join condition for the sub-query, the values are '`DynamicGlobalOR`' or '`DynamicGlobalAND`'.

- *dynamicCriteria* **Document List** Optional. A profile query used as a sub-query to filter the sender ID or the receiver ID.

  The *fieldName* variable must be either `SenderID` or `ReceiverID`. The operator value for the parent criteria must be `Dynamic`. The sub-query is added to SenderID or ReceiverID, based on what value you provided in the *fieldName*.

  If the value of *receiverEqualsSender* is `true`, then the sub-query is added to both SenderID and ReceiverID.

  For the list of variables for each document, see "Variables for dynamicCriteria variable" on page 355.

List of Column names

This table describes the field names that are valid values for the *fieldName* variables in the wm.tn.rec:queryField and the *resultSetColumns*' variable in the wm.tn.rec:queryInput IS document types. For more information, see "wm.tn.rec:queryField" on page 351 and "wm.tn.rec:queryInput" on page 356.

The column names can be:

- Any custom attribute ID of a Document Type.
- Any of the following Strings based on the query type.

Query Type...	Column names...
Document Query	■ DocTimestamp ■ DocID ■ SenderID ■ ReceiverID ■ ConversationID ■ NativeID

Query Type...	Column names...
	■ DocTypeID
	■ GroupID
	■ RoutingStatus
	■ UserStatus
	■ Comments
	■ JobStatus
	■ SenderProfileGroup
	■ ReceiverProfileGroup
Event Query	■ ActivityLogID
	■ RelatedDocID
	■ EntryTimestamp
	■ EntryType
	■ EntryClass
	■ BriefMessage
	■ RelatedPartnerID
	■ RelatedInstanceID
	■ RelatedStepID
	■ B2BUser
	■ FullMessage
Task Query	■ dj.JobType
	■ d.ReceiverID
	■ d.SenderID
	■ dj.JobStatus
	■ dj.ServiceName
	■ dj.QueueName
	■ dj.DocID
	■ dj.JobID
	■ dj.TimeCreated

Query Type...	Column names...
	■ dj.TimeUpdated ■ dj.Retries ■ dj.RetryLimit ■ dj.RetryFactor ■ dj.ServerID ■ dj.TimeToWait ■ dj.TransportStatus ■ dj.TransportStatusMessage ■ dj.TransportTime
Profile Query	■ CorporationName ■ OrgUnitName ■ GroupName ■ Status ■ City ■ State_Province ■ Zip_PostalCode ■ Country ■ Username ■ ExternalIDs ■ keywords

Valid Values for the criteria field variable

The following table lists the valid values for the *criteria* field based on the field types. The field type is always 'STRING', unless the field name is a custom attribute of a document.

Field Types	Field Values
STRING	■ IS NULL ■ IS NOT NULL ■ =

Field Types	Field Values
	■ <>
NUMBER	■ IS NULL ■ IS NOT NULL ■ = ■ <> ■ >= ■ <= ■ > ■ <
DATETIME	■ IS NULL ■ IS NOT NULL ■ = ■ BEFORE ■ AFTER
DATETIME LIST	■ IS NULL ■ IS NOT NULL
NUMBER LIST or STRING LIST	■ IS NULL ■ IS NOT NULL ■ INCLUDES

Variables for dynamicCriteria variable

The following table lists the variables for each document in the *dynamicCriteria* variable.

Value	Description
fieldName	**String** Name of the field for which you specify the sub-query filter criteria. For the list of valid values, see the values corresponding to Profile Query in the *fieldName* parameter.
isOrQuery	**String** Whether the join condition for the filter criteria in *simpleCriteria* is AND or OR. You can set the following values for the JOIN condition: ■ `true` OR

Value	Description
	■ `false` AND (Default)
simpleCriteria	**Document List** The filter criteria for a field in the sub-query. Each field can have more than one filter criteria. For example, `'GroupName IS NULL' OR 'GroupName <> G1'` where `GroupName` is the field name, `IS NULL` and `<>` are the operators, `OR` is the join condition, and `G1` is the value. Each document in the document list contains the following variables: ■ *operator* **String** The operator for the filter criteria. Valid values: ■ IS NULL ■ IS NOT NULL ■ = ■ <> ■ *value* **String** The value for the filter criteria. For example, for the filter criteria, `'GroupName < > G1'`, `G1` is the value

wm.tn.rec:queryInput

The input for a query.

Variables

pageSize	**String** Optional. The page size to use when enumerating over the query results. The default is 100.
queryID	**String** The ID of the query. This ID can be used in the wm.tn.query:getQueryResults service to get the query results.
criteria	**Document** Contains the filter criteria of the query. The document contains the following variables: ■ *dateRange* **Document** The start date and end date of the documents. Only those documents that fall within this date range are retrieved. The document contains the following variables: ■ *startDate* **String** A timestamp that filters the documents that were received after the specified value. For Java developers, this must be the *long* value obtained from the java.sql.Timestamp object.

- *endDate* **String** A timestamp that filters the document that were received before the specified value. For Java developers, this must be the *long* value obtained from the java.sql.Timestamp object.

- *isOrQuery* **String** Whether the join condition for the filter criteria that you specify in the *fields* variable is AND or OR. You can set the following values:
  - `true` - OR
  - `false` - AND (Default)

receiverEqualsSender **String** Whether the sub-query that you specify in the *dynamicCriteria* variable of the wm.tn.rec:queryField IS document type must apply to both sender and receiver. The values can be `true` or `false`. The default value is `false`.

If the value is `true` and the sub-query that you specify in *dynamicCriteria* is for sender ID, then the same sub-query is used for receiver ID too.

Similarly, if the value is `true` and the sub-query specified in *dynamicCriteria* is for receiver ID, then the same sub-query is used for sender ID too.

fields **Document List** The filter criteria for fields such as DocTimestamp, DocID. For the document structure, see "wm.tn.rec:queryField" on page 351.

sortOrder **Document List**. The column name by which the results must be sorted. Each document in the document list contains the following variables:

- *columnName* **String** The column name for sorting. For the list of valid values, see "List of Column names" on page 352.
- *isAscending* **String** The order by which the results set must be sorted (ascending or descending). Valid values are:
  - `ASC` Ascending
  - `DESC` Descending

resultSetColumns **String List** The names of the columns that must be available in the result set. For the list of valid values, see "List of Column names" on page 352.

wm.tn.rec:queryOutput

Contains the query execution details.

Variables

queryID	**String** The query ID specified in the input. You can use this ID to view the query results or to cancel the query.
resultCount	**String** Number of records in the result set after executing the query.

wm.tn.rec:ReliableServiceOutput

The output from a service that is being executed by a service execution task.

Variables

status	**String** The status the service returned. The value of *status* is either `success` or `fail`.
statusMessage	**String** The message that the service returned along with the *status*.
transportTime	**String** The total time (in milliseconds) that it took the service to execute.
output	**Document** Return information, if applicable; otherwise, null. The format is specific to the service being executed.

wm.tn.rec:SmtpDeliveryServiceOutput

The output from the SMTP (e-mail) delivery services, for example, wm.tn.transport:primaryFtp.

Variables

status	**String** The status the delivery service returned. The value of *status* is either `success` or `fail`.
statusMessage	**String** The delivery-specific message that the delivery service returned along with the *status*.
transportTime	**String** The total time (in milliseconds) that the delivery service used to deliver the document.
output	**Document** Return information from the delivery service. For the structure of *output*, see wm.tn.rec:SmtpOutput.

wm.tn.rec:SmtpOutput

If a task uses one of the following delivery services to deliver the document, you can use this IS document type to map the output from the delivery service.

- wm.tn.transport:primaryFtp
- wm.tn.transport:primarySmtp
- wm.tn.transport:secondarySmtp

Variables

status	**String** Final status from the service.

wm.tn.rec:svcResponse

Contains the response from a service.

Variables

info	**Document List**. Contains the information messages. The list contains one entry for each message. The variables are: - *key* **String** The message key. - *message* **String** The information message.
warning	**Document List** Contains the warning messages. The list contains one entry for each warning. The variables are: - *key* **String** The message key. - *message* **String** The warning message. - *stack* **String** The stack trace of the exception.
error	**Document List**. Contains the error messages. The list contains one entry for each error. The variables are: - *key* **String** The error key. - *message* **String** The error message. - *stack* **String** The stack trace of the exception.

wm.tn.rec:Task

A task.

Variables

TaskId	**String** The internal identifier for the task.

ServerId **String** The host name of the Integration Server machine to which the task is assigned.

Envelope **Document** The document that is associated with the task; that is, the document that is being delivered by a delivery task or processed by a service execution task. For the structure of *Envelope*, see wm.tn.rec:BizDocEnvelope.

Ttw **Object** If the first attempt to complete the task fails, the time to wait (in milliseconds) before making the first retry attempt to perform the task. The data type of *Ttw* is com.wm.data.MLong. (The task engine use *Ttw* along with *RetryFactor* to calculate how long to wait before making subsequent retry attempts.)

Retries **Object** If the task fails, the additional number of times the task engine attempted to perform the task (for example, deliver a document or execute a service). The data type of *Retries* is com.wm.data.MInteger.

RetryLimit **Object** The maximum number of times the task engine is to attempt to perform the task (for example, deliver a document or execute a service) after an initial failure. The data type of *RetryLimit* is com.wm.data.MInteger.

RetryFactor **Object** The factor you want the task engine to use when determining how long to wait before making the second and subsequent attempts to perform the task. The task engine calculates the time to wait by multiplying the last wait time by *RetryFactor*. Specify a whole number greater than zero for *RetryFactor*. The data type of *RetryFactor* is com.wm.data.MInteger.

Status **Integer** The processing status of the task. Valid values are:

- `NEW = 0`
- `PENDING = 1`
- `DONE = 2`
- `FAILED = 3`
- `STOPPED = 4`
- `QUEUED = 5`
- `DELIVERING = 6`
- `HELD = 7`

TimeCreated **Object** The time the task was created. The data type of *TimeCreated* is com.wm.data.MLong.

TimeUpdated **Object** The time the task was last updated. The data type of *TimeUpdated* is com.wm.data.MLong.

TransportStatus	**String** The status of the last attempt to perform the task. The value of *TransportStatus* is either `success` or `fail`. If this is a delivery task, the delivery service used to deliver the document must return the status. If this is a service execution task, the service being executed by the task must return the status.
TransportStatusMsg	**String** A message associated with the last attempt to perform the task. If this is a delivery task, the delivery service used to deliver the document returns the status message. If this is a service execution task, the service being executed by the task returns the status message.
TransportTime	**Object** The time (in milliseconds) associated with the last attempt to perform the task. The data type of *TransportTime* is com.wm.data.MLong. If this is a delivery service, *TransportTime* is the time that the delivery service used to deliver the document. If this is a service execution task, *TransportTime* is the time it took the service to execute.
InputData	**Document** Additional data required by the associated service. For a delivery task, the associated service is the delivery service used to deliver the document. For a service execution task, the associated service is the service being executed by the task.
OutputData	**Document** The data returned by the last attempt to perform the task. The data type of *OutputData* is com.wm.data.IData. If this is a delivery task, this is the data returned by the delivery service used to deliver the document. If this is a service execution task, this is the data returned by the service being executed by the task.
Service	**Document** The service that is associated with this task. If this is a delivery task, this is the delivery service used to delivery the document. For the structure of *Service*, see wm.tn.rec:DeliveryService. If this is a service execution task, this is the service being executed by the task.

wm.tn.rec:TaskDbUpdate

If the Trading Networks property, `tn.task.dbupdate.retryEnabled` is set to `true`, Trading Networks publishes a document of this type when it attempts to retry updating its database with information for a task.

This IS document type specifies the structure of the document that Trading Networks publishes.

Variables

TaskId	**String** The internal identifier for the task.
UpdateStatus	**String** Whether the attempt to update the database on this retry is successful. *UpdateStatus* is either `success` or `fail`.
UpdateRetry	**Object** If the attempt to update the database fails, this is the number of times that Trading Networks has attempted to update the database for this task. The data type of *UpdateRetry* is com.wm.data.MInteger.
TNMessage	The message that Trading Networks logs to the Integration Server log when it attempts to retry to update the database for the task. Trading Networks logs the message to the server log whether the attempt fails or succeeds. This message is logged at log level 4.
DbErrorMessage	**String** The database error message that caused the task update to fail. The value of *DbErrorMessage* is null, if *UpdateStatus* is `success`.
StackTrace	**String** If the *UpdateStatus* is `fail`, this is the stack trace of the database exception.
TaskStatus	**String** The processing status of the task. Possible values are: NEW, PENDING, DONE, FAILED, STOPPED, QUEUED, DELIVERING.
TaskRetries	**Object** If the task fails, the additional number of times the task engine attempted to perform the task (for example, deliver a document or execute a service). The data type of *TaskRetries* is com.wm.data.MInteger.
DocId	**String** The internal id of the document that is associated with the task; that is, the internal id of the document that is being delivered by a delivery task or processed by a service execution task.
TransportStatus	**String** The status of the last attempt to perform the task. The value of *TransportStatus* is either `success` or `fail`. If this is a delivery task, the delivery service used to deliver the document must return the status. If this is a service execution task, the service being executed by the task must return the status.
TransportStatusMsg	**String** A message associated with the last attempt to perform the task. If this is a delivery task, the delivery service used to deliver the document returns the status message. If this is a service execution task, the service being executed by the task returns the status message.
TransportTime	**Object** The time (in milliseconds) associated with the last attempt to perform the task. The data type of *TransportTime* is com.wm.data.MLong. If this is a delivery service, *TransportTime* is the time that the delivery service used to deliver the document. If this is a service execution task, *TransportTime* is the time it took the service to execute.

wm.tn.rec:TaskFailure

If the task failure notification feature is enabled, Trading Networks publishes a document of this type. This specifies the structure of the IS document type that is published whenever a task fails at its final retry attempt.

Variables

TaskId	**String** The internal identifier for the task.
ServerId	**String** The host name of the Integration Server machine to which the task is assigned.
DocId	**String** The internal ID of the document that is associated with the task; that is, the internal id of the document that is being delivered by a delivery task or processed by a service execution task.
Ttw	**Object** The value of time to wait parameter that the task engine used. If the first attempt to complete the task fails, the time to wait (in milliseconds) before making the first retry attempt to perform the task. The data type of *Ttw* is com.wm.data.MLong. (The task engine use *Ttw* along with *RetryFactor* to calculate how long to wait before making subsequent retry attempts.)
Retries	**Object** The number of times the task engine attempted to perform the task (for example, deliver a document or execute a service). The data type of *Retries* is com.wm.data.MInteger.
RetryFactor	**Object** The factor used by the task engine when determining how long to wait before making the second and subsequent attempts to perform the task. The task engine calculates the time to wait by multiplying the last wait time by *RetryFactor*. The data type of *RetryFactor* is com.wm.data.MInteger.
Status	**String** The status of the task.
TimeCreated	**Object** The time the task was created.
TimeUpdated	**Object** The time the task was last updated.
TransportStatus	**String** The status of the last attempt to perform the task. The value of *TransportStatus* is either `success` or `fail`. If this is a delivery task, the delivery service used to deliver the document must return the status. If this is a service execution task, the service being executed by the task must return the status.
TransportStatusMsg	**String** A message associated with the last attempt to perform the task.

	If this is a delivery task, the delivery service used to deliver the document returns the status message. If this is a service execution task, the service being executed by the task returns the status message.
TransportTime	**Object** The time (in milliseconds) associated with the last attempt to perform the task. The data type of *TransportTime* is com.wm.data.MLong. If this is a delivery service, *TransportTime* is the time that the delivery service used to deliver the document. If this is a service execution task, *TransportTime* is the time it took the service to execute.
InputData	**Document** Additional data required by the associated service. For a delivery task, the associated service is the delivery service used to deliver the document. For a service execution task, the associated service is the service being executed by the task.
OutputData	**Document** The data returned by the last attempt to perform the task. The data type of *OutputData* is com.wm.data.IData. If this is a delivery task, this is the data returned by the delivery service used to deliver the document. If this is a service execution task, this is the data returned by the service being executed by the task.
Service	**Document** The service that is associated with this task. If this is a delivery task, this is the delivery service used to delivery the document. For the structure of *Service*, see wm.tn.rec:DeliveryService. If this is a service execution task, this is the service being executed by the task.

wm.tn.rec:tpa

A trading partner agreement (TPA).

Variables

senderID	**String** The Trading Networks internal identifier for the trading partner that has the sender role in the TPA. **Note:** A TPA is an agreement between two partners in your network; one that fulfills the sender role during document exchange, and the other that fulfills the receiver role. Both the sender and receiver in a TPA must be partners in your Trading Networks system.
receiverID	**String** The Trading Networks internal identifier for the trading partner that has the receiver role in the TPA.

agreementID	**String** An application-specific identifier for the TPA.
created	**String** The time that Trading Networks created the TPA.
lastModified	**String** The time that Trading Networks last updated the TPA either when the TPA was created or last updated.
controlNumber	**String** (optional) A field that is available for application-specific use. Trading Networks does not update the *controlNumber*.
status	**String** The agreement status of the TPA, which indicates whether the TPA is a draft or final version and whether the agreement is active or not. Valid values are: ■ `Proposed` - The TPA is in draft status. ■ `Agreed` - The TPA is final. ■ `Disabled` - The TPA should not be used; it is disabled.
exportService	**String** (optional) The fully-qualified name of a service that exports a Trading Networks TPA and converts it to an industry-standard format.
initService	**String** (optional) The fully-qualified name of a service that sets default values for the IS document type defined by *dataSchema*.
dataSchema	**String** (optional) The fully-qualified name of an IS document type that defines the blueprint of the TPA, that is, establishes the TPA parameters and values.
dataStatus	**String** Whether you can update the values in *tpaData*. This field is only used when *status* is `Agreed`. Valid values are: ■ `Modifiable` - The data in *tpaData* can be modified. ■ `Non-modifiable` - The data in *tpaData* cannot be modified.
tpaData	**Document** An IData object with the structure defined by the IS document type specified in *dataSchema*. *tpaData* contains the data for the TPA.
version	**String** The version number of the TPA.
tpaID	**String** A unique ID for the TPA that Trading Networks generates.

wm.tn.rec:tpaError

An IData error object.

It is used by all APIs to report an error.

Input Variables

severity	**String** The error severity.
location	**String** The error location.
errmsg	**String** The error message.

Output Variables

None.

wm.tn.rec:TPAValidateServiceOutput

The record reference to which validation services must adhere in order to pass validation output results.

Variables

success	**String** Validation result to be passed after TPA data is validated. Valid values are: - `true` - Data is validated and the result is successful. - `false` - Data is validated but errors occurred during the validation process.
errors	**String List** If errors occurred during the validation process, pass the errors as a string array.

Usage Notes

You can use the pub.schema:validate service and map the results (validation message and errors, if any) to the output.

wm.tn.rec:WebServiceDeliveryServiceOutput

The output from the delivery service that Trading Networks provides for its web service delivery methods.

Variables

status	**String** The status the delivery service returned. Valid values are `success` or `fail`.

statusMessage	**String** The delivery-specific message that the delivery service returned with *status*.
transportTime	**String** The total time (in milliseconds) that the delivery service used to deliver the document.
output	**Document** The response returned after invoking the web service. For the structure of *output*, see wm.tn.rec:WebServiceOutput.

wm.tn.rec:WebServiceOutput

If a task uses a wm.tn.transport:webService delivery service to deliver the document, you can use this IS document type to map the output from the delivery service.

Variables

response	**Document** The required response obtained after executing the web service. The response depends on the output parameter of the web service connector in the Output Parameter field, defined in the web service delivery method. For information about defining web service delivery methods, see *webMethods Trading Networks Administrator's Guide*.
SOAP-FAULT	**Document** SOAP Fault element returned by the web service. This element is returned only when the web service does not process the document successfully. For information about SOAP-Fault see *Web Services Developer's Guide*.
wscOutput	**Document** The web service connector output obtained after executing the web service. The output contains all the data in the output pipeline, regardless of the output parameter of the web service connector in the Output Parameter field.

A Java API

Use the Trading Networks Java API to develop services or client applications. The documentation for the Trading Networks Java API is online. You can find it at:

Integration Server_directory \instances*instance_name*\packages\WmTN\doc\api\index.html