

spatialWEBSERVICES™

January 4, 2010



9635 Maroon Circle, Suite 440
Englewood, Colorado 80112
P 720.873.6880
F 720.873.6885
www.spatialinfo.com

Proprietary Information: Disseminate only with permission from **spatialinfo**. Copyright **spatialinfo** 2009

Table of Contents

Statement of Confidentiality.....	5
About This Document	6
Address Services	7
AddAddress	7
CadAddressPromoter	7
DeleteAddress.....	7
ExecuteAddressMergerAllZips	7
ExecuteAddressMergerSingleZip	7
FindAddressesForNode	7
FindCustomersForBuss	7
FindCustomersForNode	8
FindCustomersForRFPlant.....	8
FindCustomersForRFPlantByName	8
MatchCustomerToAddress	8
ParseAddressString	8
ParseAddressTable.....	9
ScanCustomer	10
Fiber Services	11
TraceFiber	11
FiberTracesInRegion.....	11
SplicingInRegion.....	11
SplicesStartingWithName	12
AnalyzeDiversity.....	12
Geocode Services	13

GetUTMForLL.....	13
GetUTMZoneForLon	13
ScanAll.....	13
ISP Services	14
FindChassisByName	14
FindCardByName	14
FindISPPortByName	14
FindCustomersForChassis	14
FindCustomersForCard	14
FindCustomersForISPPort	14
RF Services	15
CadTraceNodeBoundary	15
ConvertCADtoRF	15
ConvertCADtoRFByNode.....	15
ConvertCADtoRFRebuild	15
CreateModemSampleData_DO_NOT_RUN	15
FindAffectedCustomers	16
FindNodeByName	16
FindPlantAtBuss	16
TraceBussMPEF	16
TraceBussThreshold	16
TraceBussTrended.....	16
TraceNodeMPEF.....	17
TraceNodeTrended	17
TracePlantTrended.....	17

UpstreamTraceModems 17

UpstreamTraceModemsByName..... 17

Statement of Confidentiality

This document has been prepared under a contract between **SPATIALinfo Pty Limited (SPATIALinfo)** and the Contracting Party. The information contained in this document is confidential to **SPATIALinfo** and/or the Contracting Party and/or third parties as identified in the document and its use is strictly limited to the performance of the contract between **SPATIALinfo** and the contracting party. **SPATIALinfo** reserves all intellectual proprietary rights in the document and information contained therein Copyright © 1981 – 2008+ **SPATIALinfo Pty Limited**. The document may not be disclosed, distributed, copied, stored by electronic mechanism, either in whole or in part, or used for any other purpose other than the Contract Purpose, without the express written approval of **SPATIALinfo**. Portions of the document may contain actual material, or references to material, which is subject to Copyright © 1981 – 2008+ **SPATIALinfo Pty Limited**. No right of ownership in any pre-existing intellectual property including Copyright is transferred to any recipient of this document or of any information contained therein.

About This Document

This document contains definitions of all web services currently available in the **spatialWEB™** application server.

*Note: There are a few custom implementations of web services currently used by other customers, and the following document describes the current web services available in **spatialWEB Core 3.0**.*

Address Services

AddAddress

AddAddress creates a single dwelling address in the **spatialNET™** database from a series of input fields. All required **spatialNET** tables are populated with values.

CadAddressPromoter

CadAddressPromoter calls the CAD address promoter service that takes address blocks in the CAD tables and creates records in the CAD_ADDRESS table ready for use with the address merger service. The return is a status code value and this method for executing the service uses only a single thread/process for processing the entire database.

DeleteAddress

DeleteAddress deletes an address (a unit within a building or a SDU) from the **spatialNET** database, removing records from all internal tables as required. If all units inside a building are deleted, the service will also delete the building record.

ExecuteAddressMergerAllZips

ExecuteAddressMergerAllZips merges multiple sources of address records into a single **spatialNET** address database, including CAD, billing, business leads and subscriber records. Processing is performed within all zip code boundaries within the system using a single process/thread to merge the data.

ExecuteAddressMergerSingleZip

ExecuteAddressMergerSingleZip executes the same service as described for *ExecuteAddressMergerAllZips* but within a single zip code. If there is more than one boundary defined for a given zip code, all boundaries in that zip code will be processed.

FindAddressesForNode

FindAddressesForNode performs a trace to find all addresses connected to the node.

FindCustomersForBuss

FindCustomersForBuss takes a given node and buss and performs an RF trace of the network connected to that buss, finding all addresses connected to each tap within the plant, and then for each address finds all active customer records (active subscribers). The service operates on fully modeled RF networks, and uses the node name to perform the node search. The buss is determined by looking at the order of the modeled buss ports, where port 1 is buss A, port 2 is buss B etc. The return is a list of active customer records fed from that bus.

FindCustomersForNode

FindCustomersForNode operates the same as *FindCustomersForBuss*, but scans all busses on each node and returns a list of all customer records (active subscribers) connected to the node.

FindCustomersForRFPlant

FindCustomersForRFPlant operates the same as *FindCustomersForBuss*, but scans downstream of a single piece of equipment within the plant. The trace start points can be any of the following types (class names):

- RFActive
- RFPassive
- RFCable
- RFTap

The internal **spatialNET** ID of the plant is used to identify which object to start the trace from.

FindCustomersForRFPlantByName

FindCustomersForRFPlantByName operates the same as *FindCustomersForRFPlant*, but instead of using the **spatialNET** ID of the plant, the NAME of the RF plant is used to search for the trace start object. For example, the service can find all customers downstream of the active named "DN28A2". If two objects have the same name, an exception is thrown. Only fully modeled RF plant is considered by this service.

MatchCustomerToAddress

In the **SPATIALinfo** data model, the CUSTOMER table points at the ADDRESS table by ID. Where the customer data was sourced from the billing system subscriber records, and the ADDRESS table was sourced from a variety of places (e.g., CAD, billing, business leads), the required foreign key needs to be rebuilt. This web service correctly points CUSTOMER records at ADDRESS records using:

- The HOUSE_KEY value if a house key match can be found
- The address of the CUSTOMER record if a match can be found in the ADDRESS table

The entire database is processed in one pass.

ParseAddressString

ParseAddressString takes any input address string and parses it into the **SPATIALinfo** E911 compliant ParsedAddress structure. The input string is processed into:

- Unit (standardized to USPS format)

- Unit type
- Street number
- Directional prefix (standardized to USPS format)
- Street number
- Street name
- Street type (standardized to USPS format)
- Directional suffix (standardized to USPS format)
- City
- State (standardized to USPS format)
- Zip
- Zip+4

ParseAddressTable

ParseAddressTable takes the SOURCE_ADDRLINE column in the ADDRESS table and

- Parses it into E911 compliant addressing fields (e.g., unit, predir, street_name, street_type)
- Standardizes common misspellings of street types and states to USPS standards
- Updates the following columns in the ADDRESS table using the processed version of the source address value
 - Unit
 - Unit type
 - Directional prefix/suffix
 - Street number
 - Street name
 - Street type
 - City (if present in SOURCE_ADDRLINE)
 - State (if present in SOURCE_ADDRLINE)

- Zip (if present in SOURCE_ADDRLINE)

The entire table is processed in a single pass.

A document describing the usage of this web service is included in the standard **spatialWEB** release.

ScanCustomer

ScanCustomer searches the database for a customer record using the address as recorded in the subscriber table. It does not perform parsed address matching, so exact address matches are required.

Fiber Services

TraceFiber

Note: This feature is currently beta only and the service will change in future releases.

TraceFiber takes a fiber segment and physical position of a fiber within the fiber cable and performs a full end-to-end flood trace of the connected network. All fibers that share that light path are included in the trace results.

The results are currently implemented as a SimpleTraceNode tree of parent/child relationships, allowing both tree and mesh networks to be returned in the results.

FiberTracesInRegion

FiberTracesInRegion traces all fiber segments in a specified region.

SplicingInRegion

SplicingInRegion exports all splices in a region as a set of XML data (*SerializedFiberSplice* elements). The input argument is a boundary of type "Region" that has the given name. All splices in the region boundary are returned, regardless of where the splice is contained (e.g., ISP buildings, term racks and splice enclosures).

A *SerializedFiberSplice* contains:

- SpliceID
- SpliceName
- Parent object (what contains the fiber splice)
- GrandParent object (the ultimate OSP object that contains the fiber splice)
- CompactSplicing (a list of strings representing the summary of the splicing)
- Splicing (a list of SerializedSplice objects that provide detailed information on each fiber spliced)

SplicesStartingWithName

SplicesStartingWithName exports all splices that start with a given name as a set of XML data (*SerializedFiberSplice* elements). The input argument is a text search string that starts at the beginning of the actual splice name (not the splice enclosure name). All splices inside ISP buildings, term racks and splice enclosures will be returned by this query.

A *SerializedFiberSplice* contains:

- SpliceID
- SpliceName
- Parent object (what contains the fiber splice)
- GrandParent object (the ultimate OSP object that contains the fiber splice)
- CompactSplicing (a list of strings representing the summary of the splicing)
- Splicing (a list of SerializedSplice objects that provide detailed information on each fiber spliced)

AnalyzeDiversity

In the case where diverse, completely independent fiber routes are required, *AnalyzeDiversity* provides the ability to display diversity analysis after tracing a fiber. *AnalyzeDiversity* uses input parameters of a list of Entity(s) to return a DiversityBreach(es) report that contains:

- Breach type (i.e., Close Proximity, Crossover Point, Folded Path, Common Device, Single Lead In, Dual Lead In)
- Entities involved in breach
- Breach location (approximate)

Geocode Services

GetUTMForLL

GetUTMForLL returns the UTM coordinates for a given lat/long value. The UTM zone to use is automatically determined from the longitude. Results are in meters.

GetUTMZoneForLon

GetUTMZoneForLon returns the UTM zone for a given longitude.

ScanAll

ScanAll scans the ADDRESS table for a given address and returns the address record as a parsed address. If no address record is found, the PARCEL table is scanned for the same address record. Returns NULL if no address is matched, and throws an error if more than one address matches the search criteria.

ISP Services

FindChassisByName

FindChassisByName finds an ISP chassis by ID for a given building name and chassis name combination. It returns NULL if either 0 or more than 1 chassis are found.

FindCardByName

FindCardByName finds an ISP card ID for a given building name, chassis name and card name combination. It returns NULL if either 0 or more than 1 cards are found.

FindISPPortByName

FindISPPortByName finds an ISP port's ID for a given build name, chassis name, card name and port name combination. It returns NULL if either 0 or more than 1 ISP ports match the search criteria.

FindCustomersForChassis

FindCustomersForChassis finds all customers fed downstream from a given ISP chassis. The system finds all cards in that chassis, then all ports on each card, then uses the non-standard ISP/OSP port mapping table to match these ISP ports to RF node buss ports in the OSP network. The system then traces each buss port to return the list of affected customers. The return result is a list of all customer records.

FindCustomersForCard

FindCustomersForCard operates the same as *FindCustomersForChassis*, but operates on a single card.

FindCustomersForISPPort

FindCustomersForISPPort operates the same as *FindCustomersForChassis*, but operates on a single ISP port on a single card.

RF Services

CadTraceNodeBoundary

CadTraceNodeBoundary traces the CAD graphics for RF symbols and lines inside a node boundary and creates a trace result of the CAD as a trace tree. It uses the configuration stored in RF_EQUIP_DICT as edited using the CAD to RF configuration tool in **spatialWEB**.

ConvertCADtoRF

ConvertCADtoRF traces the CAD graphics for RF symbols and lines inside a node boundary and creates fully modeled RF plant in a **spatialNET** database (nodes, actives, passives taps and coax cables). The modeled network does not have design or level information available.

Addresses are connected to taps as part of this process, using the CAD connection attribute (e.g., LOC_NUM) if this value is available and within 400ft of the tap, otherwise addresses are connected to taps using a proximity scan.

Nodes are upgraded by name (as stored within the CAD node block). A modeled node boundary with the same name as the node must also be stored in the system.

If a node has previously been promoted, the system will delete and recreate the modeled RF data.

All nodes are rebuilt by this service.

ConvertCADtoRFByNode

ConvertCADtoRFByNode performs the same trace and promotion as *ConvertCADtoRF*, but on a single node. The node is specified by its node name (as stored in the NODE_ID or NAME attribute of the CAD node symbols).

ConvertCADtoRFRebuild

ConvertCADtoRFRebuild currently works identically to *ConvertCADtoRF*. In future revisions it will be modified so that this service rebuilds all nodes and the default *ConvertCADtoRF* node will only rebuild nodes if the CAD data has changed within that node boundary.

CreateModemSampleData_DO_NOT_RUN

CreateModemSampleData_DO_NOT_RUN creates sample data for use in testing the historical modem analysis software, which in a production system is populated from the CMTS records. This service will delete all existing modem statistics data and should only be used in a test system.

FindAffectedCustomers

FindAffectedCustomers returns a list of affected modems (including customer details) downstream of a given trace start point in the RF plant. A customer is considered "affected" if one or more of the devices they own has a status code that we ask the web service to scan for (e.g., find customers whose modems are listed as "OFFLINE" or "FAILED" downstream of a specific active).

FindNodeByName

FindNodeByName finds the ID for a modeled RF node based on the name of the name.

FindPlantAtBuss

FindPlantAtBuss finds the piece of plant attached to a buss on a node, and returns the plant class and id. The node is identified by node name, and the buss is identified by the port number of the port on the node (e.g., port 1 is buss A, port 2 is buss B).

TraceBussMPEF

TraceBussMPEF performs a root cause analysis on a buss to determine if one or more pieces of plant have failed on the buss. A piece of equipment is determined to be a root cause if all modems downstream of the equipment are "failed" and modems upstream in the network are "ok".

Taps are considered OK or failed if a percentage of modems at the tap have one of the requested status codes.

The percentage required to call the tap "failed" is configurable.

TraceBussThreshold

TraceBussThreshold performs an RF trace of the buss and returns a list of RF network elements where a modem on that element first meet certain conditions. Any plant downstream of where the condition was first met will not be considered.

A list of conditions can be used. If any of the conditions specified are met for any modem at a network element, the network element will be selected and trace downstream will stop for that branch.

TraceBussTrended

TraceBussTrended performs an RF trace from a given buss port, and finds all modems downstream of the buss. For each modem, the most recent modem statistics/telemetry information is read from the history of modem statistics data.

Telemetry data is then aggregated back up the network using the RF plant trace information, creating a weighted average of network performance for every location on the network tree.

The return value from this service is the weighted aggregate of modem statistics for all modems downstream of a buss port.

TraceNodeMPEF

TraceNodeMPEF performs the same analysis as described in *TraceBussMPEF* but on the entire node.

TraceNodeTrended

TraceNodeTrended performs the same analysis as *TraceBussTrended* but the analysis is performed on the entire node as the trace start point.

TracePlantTrended

TracePlantTrended performs the same analysis as *TraceBussTrended* but the analysis is performed downstream of a single piece of plant in the RF network only.

UpstreamTraceModems

UpstreamTraceModems starts at a given point in the RF network and traces the network upstream back to the node, creating a list of all modems connected to that route back to the node.

The inputs to this service are a class name and ID, and the return value is a list of modems.

UpstreamTraceModemsByName

UpstreamTraceModemsByName operates the same as *UpstreamTraceModems*, but the trace start point is identified by class name and equipment name instead of by ID. The service will error if more than one piece of plant has the same name.