iSpatial Version 3

TECHNICAL WHITEPAPER

WWW.T-SCIENCES.COM

1555 Wilson Blvd.
Suites 125, 504 and 701
Arlington, VA 22202
703-740-8768

INTRODUCTION

The concept of geo-literacy has taken hold throughout both the Government and Commercial sectors. Enterprise information management systems have historically shown data in a way that lacked a rich visualization capability or provided a map view that was not dynamically connected to a data store. The field of Geospatial Information Systems (GIS) was specific to a group of subject matter experts (SMEs) and the tools available to them were not meant to be utilized in day-to-day operations at an Enterprise level. The advent of web-based tools like Google Maps, Bing, ArcGIS, and Google Earth have dramatically changed the landscape as the average user is now well-versed in managing data in these 2D or 3D geospatial formats

iSpatial was originally designed as a geospatial framework to rapidly deliver geo-functionality to any user with minimal training required. Intended as an "80% solution for hundreds of problem sets," iSpatial exists as a set of ready-to-customize, baseline tools that can be rapidly adapted to meet nearly any use case calling for geo-visualization. These "common denominator" components were identified through exposure to many different use cases and recognized as critical to most engagements. The software regularly incorporates new and more streamlined components over time, with the internal architecture updated to leverage these new advances. With the release of iSpatial version 3, the latest features include an updated user interface, native ingestion, mobile device integration, enhanced search capability, and a enhanced capabilities for indexing of geo-objects.

As iSpatial continues to improve, it is enhanced by its user base and their interaction with the software. This establishes an economy of scale around each feature within the software and the iSpatial Product Team streamlines each component based on user feedback. This practice decreases customer costs over time as users continue to benefit from product enhancements. Combined with iSpatial's modular approach to customization, an advanced geospatial system is possible within a short period of time simply from the product of iSpatial, Google Maps or Earth, and light customization.

In contrast to being developed as an end-state capability that is requires specific knowledge of the product's core, iSpatial is an open software framework. This means that the Application Programming Interfaces (APIs) contained within it, the software libraries, and the customized toolsets can be readily accessed by any developer with the right skill set. These APIs allow an organization to springboard off the existing framework and develop the specific features they need for their use case. These customizable areas of iSpatial are available as a Software Development Kit (SDK). Organizations have been able to go from beta to production with iSpatial in as little as two to three weeks using this SDK.

iSpatial consists of four major areas of core functionality: Authoring, Searching, Managing, and Collaborating. It is a front-end browser-based interface developed in JavaScript and ExtJS, a collection of REST services that connect the user interface to the back end, and a PostgreSQL/PostGIS and MongoDB back end. Oracle Spatial, Google Map Engine, and Fusion Table extensions can be integrated, as needed, depending on the deployment model. The software requires Google Maps for Business/Google Earth Server, Postgres/PostGIS 9.1.x or higher, Python 2.7.3, and RedHat Enterprise Linux or CentOS Unix 5.8 (6.4 preferred) for server support. Users require only the free Google Earth browser plug-in and Internet Explorer 8.0+/Firefox 17+/Chrome 21+.

OVERVIEW

iSpatial is a geo-visualization framework that enables users to easily build Web-based solutions to author, manage, and collaborate within their secure enterprise over the browser-based 3D Google Earth or 2D Google Maps. iSpatial was designed to give an organization a complete framework to rapidly take new and existing data – no matter what system it is in – and represent it via a Web interface, geospatially. iSpatial deals with static legacy data as well as real-time operational data such as current locations of vehicles/cell phones, future planned activities, simulation data, and tour/fly-through data.

Figure 1 Basic iSpatial Interface

iSpatial is comprised of an enterprise geo-store, a geo-index, a collection of geo-services (REST/JSON), Web visualization libraries (JavaScript), and core user interfaces. iSpatial is tightly coupled with the Google Earth/Maps Enterprise suite of capabilities and natively connects to the Google Earth Web browser plug-in and the Google Maps Web browser interface. Its primary use is to geo-enable existing or new Web applications over Google Earth or Maps in a Web environment, maximizing compatibility across platforms.

A key strength of iSpatial is its scalability, which allows thousands of users to simultaneously access it via the Web. It is designed to run in either a traditional hosting or cloud environment with a back end that processes millions of geo objects, displaying tens of thousands of objects at one time in a browser, conducting near real-time refresh of the data being ingested and displayed, and supporting thousands of concurrent users.

iSpatial was made available for Government use as an alpha release for the first time in 2008 for the U.S. Department of State's Bureau of Diplomatic Security. It was rapidly adopted as a core component of their Next Generation Blue Force Tracking system, a system used to track the locations of diplomats stationed in hostile areas. The software gained further operational Government use when it was selected as the framework for Operation Unified Response (OUR) as

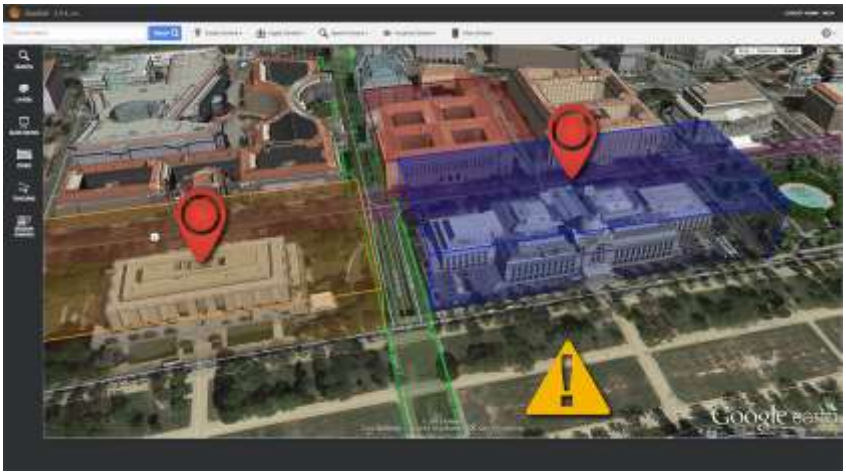


Figure 2 iSpatial visualizing different sized areas of interest

the United States and other nations responded to the 2010 earthquake in Haiti. While initially designed for commercial use, the Federal Government's strong interest in the framework led to organizations like the Defense Advanced Research Programs Agency (DARPA), Army G2, and national intelligence agencies leveraging it in their enterprise systems. It has since been deployed to multiple commercial customers as well, including NBC Universal, Willbros Group, TowerCo, and others. Many customers use an enterprise licensing approach, with the largest to-date occurring when the U.S. Army Military Intelligence (G2) community procured an enterprise license to leverage the framework for their core web-based geo-visualization. An increased effort has been made since that time to expose all iSpatial functionality for commercial use, which has created a broad user community and economy of scale when building out new features or fixing bugs.

CORE FUNCTIONALITY

iSpatial is comprised of 4 major areas of functionality: Authoring, Searching, Management, and Collaboration. These 4 core feature areas represent the foundational capabilities of the framework, but do not represent limits on its extensible potential.

AUTHOR

iSpatial provides users the ability to create content directly on a globe/map. Discussed in greater detail further down, the ability to author data includes creation, annotation, and layering of data to create robust, custom data sets within the browser. The drawing of points, lines, polygons, corridors, and other standard shapes is supported. Aside from manual creation, iSpatial features a one-click ingestion process to incorporate external data files/sources into the map. This ability allows for the layering of multiple data sets from multiple sources.

SEARCH

iSpatial's search capability is integrated directly into the framework which allows for native searching of global data that has been associated with the system. The search ability allows for keyword search, geospatial boundary-defined search, temporal search, and other options. Search results can be grouped and manipulated directly from the search results box.

MANAGE

Roles, rules, and data management capabilities are all native to iSpatial. Rights and roles management for user-level access are available to define data, approvals, and other functions for different user groups. Geospatially-defined boundaries can be set with rule sets, alerts, and notifications. As assets move in and out of these zones, rule sets can be set to trigger any number of reactions from mobile device access to priority alerts. Data management features both manual and automatic options for updating information. Aside from the standard data manipulation for updating individual fields, boundaries, metadata, etc., iSpatial can connect directly to live data feeds, bringing the latest version of the data as often as it is updated. This is especially helpful when using data that changes frequently such as weather, asset locations, and inventories.

COLLABORATE

Strong collaboration features have been a part of iSpatial since its initial development. Sharing data, data sets, and data layers is possible through simple export functions as well as a URL-sharing feature to allow non-iSpatial users to view the very same data. Language support options allow for the user interface to swap between over 50 languages supported by the integrated translation service. Finally, real-time collaboration combines screen-sharing with live sessions so users of iSpatial can interact with data while simultaneously participating in a live presentation, controlled by another iSpatial user anywhere in the world.

TECHNICAL IMPLEMENTATION

WEB STACK

The iSpatial backend is written in Python, a concise programming language that supports multiple inheritance (mixins) and is endorsed by such tech giants as Google. Multiple inheritance is a key concept in the iSpatial architecture, enabling a very simple means of adding complex functionality to objects. You can see how we take advantage of multiple inheritance in the Search Documents section, but it's important to note that anybody looking to build an application on top of iSpatial using the SDK will also be able to take advantage of multiple inheritance. See the Extending iSpatial section for more information.

iSpatial uses the Django web framework to provide an easy MVC (Model, View, Controller) framework upon which to build the application. Datastore models and RPC endpoints are written in Python, and take advantage of the libraries Django offers. The View component is a single page running an ExtJS web application which handles all frontend transactions.

JAVASCRIPT FRONT END

The iSpatial Javascript frontend leverages the [Sencha](http://www.sencha.com) ExtJS framework. In addition to supporting mixins, ExtJS offers an easy set of tools for building UI components quickly, including grids, trees, and menus. Furthermore, while other Javascript frameworks are designed around the concept of multi-page sites, ExtJS excels at emulating a single-page, desktop environment-like applications, such as iSpatial.

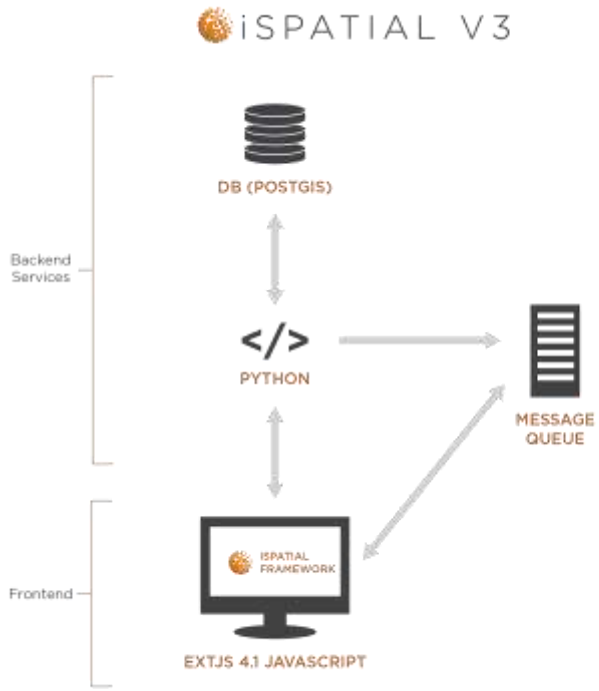


Figure 3 iSpatial technical structure

CORE SERVICES & CROSS-PRODUCT INTEGRATION

There are many components of iSpatial, such as Tracking, Zones, and Session Sharing, that rely on fast, lightweight, real-time data transport. While there are message buses and brokers that already exist on the market, Thermopylae Sciences and Technology needed a low-latency, high performance message queue for its Ubiquity platform that was capable of saving battery life and bandwidth when communicating with mobile devices.

iSpatial relies on TST's MessageQueue broker for all of its features requiring real-time interaction with users on the front-end. The choice to dogfood our own products was simple:

- MessageQueue provides out-of-the box integration with Ubiquity applications, making it very easy to interact with mobile devices from within iSpatial
- MessageQueue is fast and lightweight, so there are no performance bottlenecks
- MessageQueue is capable of scaling to hundreds of thousands of simultaneous connections, so future front-end scaling requirements are met.

By using the [Atmosphere](https://github.com/Atmosphere/atmosphere/wiki/jQuery.atmosphere.js-API) plugin, iSpatial is able not only to push data to connected users in real-time, but also allow users to directly broadcast information to one another without having to also go through the backend. This allows for smooth performance in such features as Session Sharing, and offloads processing to connected clients (rather than rely on the backend). As a result, iSpatial is capable of scaling to a large number of simultaneously-connected users, provided the users' browsers are capable of supporting [WebSockets](http://en.wikipedia.org/wiki/WebSocket).

For more information on [Ubiquity](http://ubiquity.t-sciences.com) and the Core Services offerings of Thermopylae Sciences + Technology, please see their respective whitepapers, or visit http://t-sciences.com.

DATA STORE

The goal of the data store in iSpatial is to be able to index geospatial data for easy search and retrieval alongside other relevant information. It is important that data be indexed in three dimensions, and that all associated information be easily retrieved alongside the object.

We evaluated multiple technologies when considering potential data stores, but settled on using PostgreSQL with the [PostGIS](http://postgis.refractions.net/) plugin for building iSpatial. While other Geospatial indexes only support points or simple 2D polygons with bounding-box as the only type of search criterion, the PostGIS plugin meets all of iSpatial's advanced geometric requirements, with [sufficient capabilities](http://postgis.refractions.net/documentation/manual-1.5/reference.html) to ensure forward-compatibility with future geospatial features.

The relational nature of a well-normalized data store also fits well with iSpatial's information-sharing paradigm. The idea that objects can belong to one or more layers, that users can belong to one or more groups, and that boundaries and targets have rules that comprise zones, all lend themselves well to a relational data store. Additionally, PostgreSQL supports industry-standard data store practices, including ACID transactions, the ability to support data migrations, data replication, and more.

Going forward, the iSpatial team is exploring the possibility of using non-relational data stores to provide extreme scalability for geospatial data and analytics.

INGEST

iSpatial supports the ability to ingest geometric data from a wide variety of formats and index them its database. This confers the unique advantage of allowing users to consolidate data from disparate data sources and allow them to perform analytics and intelligence gathering across multiple data sets.

Right now, the iSpatial ingest process supports the following popular formats:

- `KML`/`KMZ`
- ESRI Shapefiles (uploaded as a `ZIP` file with the extension `.shp.zip`)
- `GPX`
- `CSV`

Figure 4 iSpatial interface supporting a KMZ format

Because the iSpatial ingest process utilizes the GDAL/OGR open source libraries, however, we have the potential capability to support upwards of 50 different file formats for ingest.

The ingest service was designed to process large data sets without impacting system performance. Outlined below are the technologies used to accomplish this.

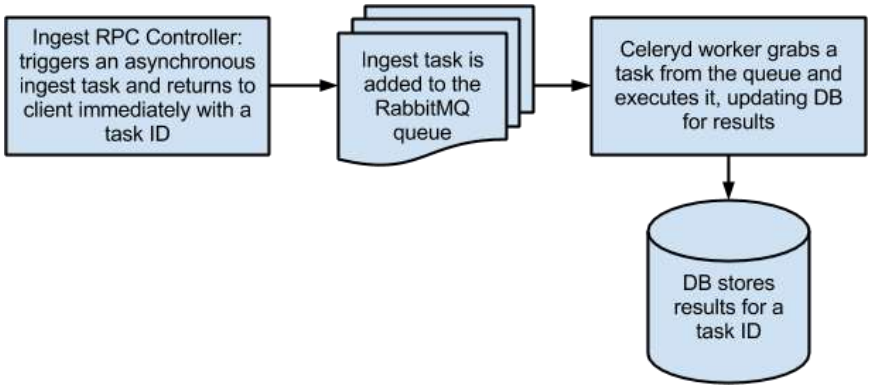
CELERY TASK QUEUE

Task Queues are used as a mechanism to distribute work across multiple threads or machines. A task queue's input is a unit of work (a task). Dedicated worker processes are constantly monitoring the queue for new work to perform.

A Celery system can consist of multiple workers and brokers, giving way to high availability and horizontal scaling.

iSpatial uses the [Celery Task Queue](https://pypi.python.org/pypi/django-celery) to register ingest requests as tasks for dedicated ingest workers to parse and save to the database. Rather than ingest data synchronously, which would tie up an HTTP connection while the ingest was taking place, using a task queue for the ingest service enables iSpatial to process large volumes of data in a short period of time while maintaining data integrity (synchronous file processing is susceptible to lost packets and dropped connections).

The following diagram gives a high-level overview of role a Task Queue plays in the ingest process.



RABBITMQ (AMQP)

Celery communicates via messages using a broker to mediate between clients and workers. To initiate a task a client puts a message on the queue, the broker then delivers the message to a worker.

The Celery Task Queue uses [RabbitMQ](http://www.rabbitmq.com/), an [AMQP](http://www.amqp.org/) implementation, as its message broker to communicate between workers and brokers for ingesting files. There are numerous advantages to using RabbitMQ as a broker over virtual transports or a database as the ingest broker, especially when it comes to preserving data in the event of abrupt power failure; see the [Celery Documentation](http://docs.celeryproject.org/en/latest/getting-started/first-steps-with-celery.html) for more information.

RabbitMQ is feature-complete, stable, durable, and easy to install, making it the ideal broker for iSpatial's ingest process.

GDAL/OGR

In order to support a wide variety of ingest formats, iSpatial relies on the open source [GDAL/OGR](http://www.gdal.org/) library to parse uploaded files.

GDAL is responsible for identifying the layers and objects present in the uploaded files, and extracts the title, description, geometry, and other fields for every object contained therein. The worker responsible for parsing the file saves each object individually to the iSpatial database, allowing for Django to execute the proper signals against that object (such as Search Document generation; see the Search Algorithm section below for more information).

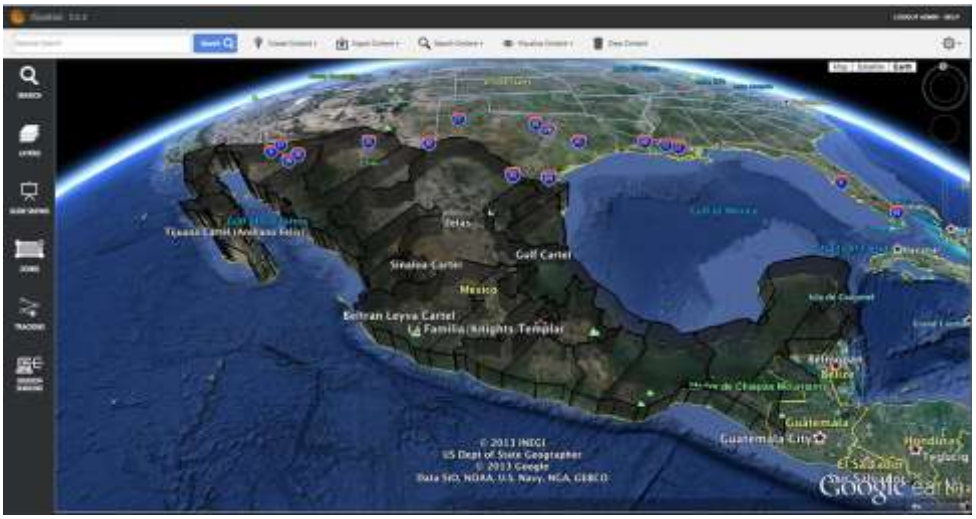


Figure 5 GDAL identifying layers in iSpatial

EMAIL NOTIFICATION

Because the iSpatial ingest process is an asynchronous process, it is important to notify users when their files have completed undergoing the ingest process. iSpatial accomplishes this by using the `django.core.mail` service to notify users via email when the iSpatial ingest process has completed parsing the file.

This email notification will also inform users as to whether or not their ingest was successful (for example, in cases where the file uploaded contains incorrect syntax).

SEARCH ALGORITHM

iSpatial's search algorithm takes the best of language parsing offered by such technologies as Apache Lucene, and combines it with advanced geospatial indexing, giving geospatial context to your data without compromising the types of features you would expect from a modern Search application. We accomplish this through the generation of "Search Documents" that index data whenever it is stored in the iSpatial database. The end result is the ability to search for content geospatially as well as by arbitrary attributes, including the use of comparison operators. For example, "show me all commercial real estate on Wall Street above the 34th

floor where the rent is less than $100/sq.ft" takes about as long to query in iSpatial as it does to read that statement out loud. Let's take a look at the technologies that enable us to accomplish this.

LANGUAGE STEMMING

Stemming is the process of reducing inflected (or sometimes derived) words to their stem, base, or root form. An example would be identifying the strings "cats," "catty," and "catlike" as having the root "cat." This is an important function of most search engines, because it allows users to identify the content they seek even if they're unable to recall the exact name of their object.

Using Python's NLTK (Natural Language Toolkit), iSpatial provides language stemming for 14 different European languages (for which the process of stemming is relevant), configurable in `settings.py`. This means that, even if you're not storing English-language data in your database, you can still identify points belonging to the "Forestry Service" simply by searching for "forest."

Figure 6 Content Search

SEARCH DOCUMENTS

Search Documents, in addition to containing indexed (stemmed) text for all fields in the object, contain the object's geometry, timestamps, and all identifying information for accessing it in the database. This allows us to construct complex queries incorporating all object properties while simultaneously filtering objects based on geometric conditions.

Search Documents are created whenever an object declared with the `Searchable` mixin is saved in the database. The `Searchable` mixin is responsible for determining whether a search document exists already for the object being saved, for creating the search document if it doesn't, and updating the search document as appropriate.

The Search Documents have a distinct advantage over Lucene and other string-based search algorithms because it enables iSpatial to perform geometric searches using the indexed geometry in PostGIS, rather than use Lucene to perform an inaccurate bounding-box search, or to use Lucene for text searches and then have to conduct a second query to bound the results geometrically.

Using a proprietary storage scheme, we automatically detect and store the type of data in search queries and object attributes (e.g., `100` is an integer, `03/02/2012` is a date, etc.). Unlike standard text-based search solutions, this allows fast and accurate typed comparisons, such as data ranges or integer comparisons. For example, you could quickly retrieve all properties where `height > 150` and `build_date >= 01/01/2013` and `contractor = "Big Cement Corp."`.

SEARCH ALGORITHM

At a high level, iSpatial's search algorithm retrieves data in the following manner:

- Receive information from the front-end
  - Does the search have a geometry?
  - Does the search have a start date?
  - Does the search have an end date?
  - All text
- Parse text to determine if there are expressions (for example, `population > 200`)
- Build a query using Django's DB tools:
  - Geometry INTERSECTS (encompasses "within") provided geometric search criteria
  - `object_datetime >=` provided start date
  - `object_datetime <=` provided end date
  - Join each search term (`AND` together; all search terms (stemmed) must be contained within the Search Document's list of stemmed keywords
  - Apply derived expressions as object attributes filters
- Execute query against Search Documents table, joining object IDs with the object table to pull back complete records and allow for attribute filtering
- Pass Objects to the front end

In this way, we need only conduct a single query against the database to retrieve object data while accounting for language parsing, geometric and temporal binding, and advanced query expressions.

EXTENDING ISPATIAL

One of the fundamental features of iSpatial is the ability to extend its core functionality and implement custom features on top of the base framework. iSpatial strives to make this as simple as possible by providing an SDK which contains unminified Javascript and scaffolding to create iSpatial applications. Detailed instructions are provided with the SDK, but the premise remains simple.

SCAFFOLD

The first step to creating an iSpatial application is to invoke the `init_app` scaffolding command. This command creates a skeleton iSpatial application in the specified directory, complete with the appropriate directory structure, files, comments, code samples, and a custom settings file.

If you opted for the default directory when running the `init_app` command, then your application is already included in your iSpatial installation! If you opted to install your app in a custom location, registering your application with iSpatial is as simple as creating a symlink in the `ispatial` directory and bouncing your web server.

IMPLEMENT NEW FUNCTIONALITY

Because your application is included as part of the iSpatial framework, you have access to all of iSpatial's core functionality that you can extend as you see fit.

EXAMPLE: CUSTOM CONTEXT MENU ITEM

Let's say we're making an application that adds Google Search Appliance integration into core iSpatial. We've

scaffolded a project "`gsa`", and now we need to add a new menu item to our right-click context menu. By overriding the default `tst.ispatial.object.Metadata` class, we create a new `gsa.object.Metadata` class that has the same context menu as base iSpatial, but with one extra option. See the following example:

```javascript
/**
 * This overrides the normal Object COM behavior to add a special
 * context menu item for google search appliance.
 **/
Ext.define('gsa.object.Metadata', {
    extend: 'tst.ispatial.object.Metadata',
    requires: 'gsa.search.SearchApplianceInterface',
    getCOMContextMenuActions: function(){
        var actions = this.callParent(arguments); // Inherit parent context menu
        actions.push( this.getSearchApplianceMenuAction() ); // Add new item
        return actions;
    },
    getSearchApplianceMenuAction: function(){
        return this._contextMenuItem( 'search_appliance_query',
            this.searchApplianceQuery, [], 'searchWithinIcon', 1, 10 );
    },
    searchApplianceQuery: function(queryText){
        // Custom function implemented in another class that interacts with GSA
        gsa.search.SearchApplianceInterface.queryForCOMs(this);
    }
});
```

By providing access to iSpatial's existing functionality and features, more is accomplished in fewer lines of code, and new features blend cleanly into the core application.

SUMMARY

iSpatial grants organizations a complete geospatial framework for rapid representation of data via the Google Earth browser plug-in and the Google Maps browser interface. iSpatial handles static, legacy data as well as real-time operational data with simulation, fly-through, planning, and other activities. This near-real-time (NRT) asset tracking and communication allows organizations to allocate resources effectively and efficiently.

iSpatial is designed to be scalable and can run in a cloud environment that can be capable of processing millions of geo objects, displaying thousands of objects at one time in a browser, conducting NRT refresh of the data being ingested and displayed. Support for thousands of concurrent users is also possible with the proper infrastructure.

As a framework, iSpatial is designed to allow for rapid development of tools and capabilities on top of the software. The SDK makes this possible, with custom software abilities developed and integrated into an iSpatial instance. This makes iSpatial an ideal tool for highly specialized requirements and missions. It gives users the functionality and flexibility they need, when they need it.