SF*park* Availability
Service API Reference

Prepared By:

Oracle Consulting

Last Updated:
April 19, 2011

CIRCLE LESS, LIVE MORE

SFMTA

Table of Contents

1 Introduction

Data is the cornerstone of the San Francisco Municipal Transportation Agency (SFMTA) SF*park* project. The SF*park* project is largely focused around a real-time feed into an SFMTA hosted data store (part of SF*park* Data warehouse) for physical parking activity as collected by sensors installed for on-street metered parking spaces and gate controlled off-street (OSP) facilities. The data from sensors focus on updates to the status for each space as "occupied" or "vacant" or occupied count for OSP facilities. Any other applicable operational schedule/ closure data, taken together and applied to these individual real-time data at any point in time are used to calculate if "vacant" spaces are in fact available for general parking.

The SF*park* data warehouse will process the real time data and provide the basis for the Parking Availability (or simply availability) feed. This real time SF*park* availability feed is being provided as a REST service with data returned in XML or JSON format.

This document includes API information needed by SF*park* API end users and partners to effectively carry out testing for retrieving and displaying the Availability data from the SF*park* Data Warehouse. It is intended to be a living document, updated as required during the life of the project. Use of the data and the API is governed by the SF*park* rules and limitations as applicable. For latest information on the project and other relevant information, including developer resources, please visit http://sfpark.org

1.1 Objectives

The main objective of this document is to serve as a reference that describes the SF*park* availability services API, their format and associated parameters that are necessary to carry out connectivity and testing of these services.

This document also provides basic information to set up the service client along with the connection properties for successfully connecting to and passing the request parameters for obtaining the availability data from the SF*park* system. It should however, not be used to determine the type, level of testing or the validity of the request parameters to be passed.

The formats of valid messages, success and error responses are also described for determining whether the service was invoked successfully or resulted in errors. It however, does not provide information on connection errors that may be due to

network or firewalls. In general once the connectivity to the SF*park* system has been established, the error response from the service should provide useful information to determine the cause or reason of the error.

The guide also provides basic steps to configure a popular Web Services Testing tool, SoapUI for testing the service. Since there are variety of ways to test such a service and a number of tools that can be used, this guide does not attempt to provide how to resolve issues with setting up any particular service client.

1.2 Special Considerations

Key aspects of the SF*park* system are discussed in this section. It provides information regarding certain characteristics and or conventions that are used by the SF*park* system.

The data provided in the response is based on the actual status events as received from the various sensors and may have changed since the events were transmitted from the individual spaces and stored and processed prior to returning this data. Hence, the status provided in the response has some lag and may provide a different view than actual status. The real time status refers to state of the data as available from the SF*park* data warehouse at the time of request.

1.2.1 Using the XML or JSON response format

SF*park* Availability service can return data in XML or JSON format (along with JSONP support) for increased flexibility to display the results as appropriate. JSON is becoming increasingly popular; however, XML is also used extensively when dealing with structured data. User thus has the choice to decide which format to use based on the merits of each format, availability of tools or developer resources for a particular format and the suitability for the target display or usage.

1.2.2 Note Regarding DNS Names/ IP Addresses used in this guide

The service should be accessed by utilizing the current SF*park* service DNS name/port number. This will avoid connectivity problems in case the associated IP address(es) for the DNS name change, which are subject to change by SF*park* at any time.

For illustrative purposes where hostname and or port numbers are needed this document references the current DNS name for the SF*park* server: api.sfpark.org running on default HTTP port 80. If the DNS server/port information changes in the future or if a secondary environment is provided for testing, then replace these connection parameters with the appropriate server/ port information.

1.2.3 SF*park* Data Conventions

Following are some of the key data conventions adopted by SFpark:

1. XML element names are specified as UPPERCASE with '_' allowed for readability. Example: `<MESSAGE>25 records found</MESSAGE>`
2. JSON response data elements are similarly also specified as UPPERCASE, e.g., MESSAGE : `"25 records found"`
3. JSON request parameters are allowed to be case insensitive.
4. All prices returned from the service are in US Dollars.

1.2.4 SF*park* Date-Time Conventions

In general, date-time elements returned from SF*park* and the SF*park* Data Warehouse follow the Pacific Time zone (PST/GMT-08:00 or PDT/GMT-07:00) in effect based on the time of the year unless otherwise noted.

Also, the response elements representing timestamp data will be formatted as per the XML dateTime convention and will provide the applicable offset from UTC/GMT, e.g., PDT offset is specified as -07:00. Hence the dateTime: "2011-04-18T12:28:41.838-07:00" represents "2011-04-18T12:28:41.838 PDT" or "2011-04-18T19:28:41.838Z" where Z indicates UTC (or GMT which is 7 hours ahead of PDT) time representation. Similarly, when SF*park* follows the PST time zone, the response will be provided using the offset: -08:00 which is equivalent to PST in numerical representation and this offset represents a time being 8 hours behind UTC.

The time element itself is incomplete without the time zone (or offset) and hence the users of the data should always consider the offset (when provided) to derive the actual time it represents. Such usage will allow for correct time interpretation in cases when the offset or the time zone changes.

When not representing a timestamp value and the response refers to certain time of day the usual SF*park* convention is to use 12 hour AM/PM time format. As above, these will represent the time for the SF*park* time zone in effect (PST or PDT). E.g., when providing the time interval for a particular price in effect: 7:00 AM – 6:00 PM: $2.00 Per Hour.

1.2.5 Acronyms and Abbreviations

Following are some of the common abbreviations and acronyms used throughout this document.

Term	Definition
API	Application Programming Interface
JSON	JavaScript Object Notation. It is a lightweight data-interchange format.
JSONP	JSONP or "JSON with padding" is a complement to the base JSON data format.
REST	Representational State Transfer. The REST protocol is a simple HTTP-based protocol.
SFMTA	San Francisco Municipal Transportation Agency.
SFpark	SF*park* is a SFMTA project. It allows for new parking management approaches and technology to manage parking supply and demand more intelligently.
SOA	Service Oriented Architecture.
XML	Extensible Markup Language (XML) is a simple yet very flexible text format.
XSD	XML Schema Definition is an XML-based language used to describe and control XML document contents.

2 SF*park* Availability REST Service API

This section provides API information for the SF*park* REST based Availability Service (or simply service). This SF*park* service provides the mechanism for obtaining the latest Availability data from the SF*park* Data Warehouse.

2.1 *REST based Availability Service API*

It is recommended for the users of the service to implement suitable response status, error checking and recovery and handling of any planned or unscheduled maintenance of the SF*park* services. Testing should also help understand the response status for various common kinds of errors that may be expected during such transmissions. e.g., network latency and or errors, incomplete or invalid format of request parameters, etc.

2.1.1 Service Endpoint

Following is the information for connecting to this service and can be tested by passing appropriate Availability Request (or filter) parameters. The service allows these parameters to be case in-sensitive. The parameter names and values should conform to the rules of this service. For the specific rules governing these parameters and details on the various search parameters allowed by the service, please review the section "Availability Search Parameters".

The REST availability service endpoint URL is:
http://api.sfpark.org/sfpark/rest/availabilityservice

Note: The URL portion above is case-sensitive. Any parameters beyond the URL endpoint are case-insensitive.

2.1.2 Sample request for testing

The service will validate the request parameters passed as query string beyond the above endpoint. Example below shows the query parameters which could be passed in using a browser or set up in an appropriate REST service test tool such as SoapUI that is described in this document. Note that testing tools may require specifying the parameters in a different manner.

http://api.sfpark.org/sfpark/rest/availabilityservice?lat=37.792275&long=-122.397089&radius=0.25&uom=mile&response=json

The service will return a SUCCESS with availability data or ERROR response based on the outcome of the request processing. Optional request parameters may be omitted in the request in which case the service will use its internal default values as appropriate for those parameters.

Upon successful invocation the service provides a HTTP 200 status code along with a SUCCESS message response. A sample response for successful invocation of the service is provided below:

```
HTTP/1.1 200 OK
Date: Thu, 17 Feb 2011 02:05:08 GMT
Content-Length: 7981
Content-Type: application/json; charset=utf-8
X-Powered-By: Servlet/2.5 JSP/2.1

{
    "STATUS": "SUCCESS",
    "NUM_RECORDS": "52",
    "MESSAGE": "52 records found",
    "AVAILABILITY_UPDATED_TIMESTAMP": "2011-02-16T18:05:08.240-08:00",
    "AVAILABILITY_REQUEST_TIMESTAMP": "2011-02-16T18:05:08.180-08:00",
    "AVL":      [
            {
        "TYPE": "OFF",
        "OSPID": "934",
        "NAME": "Golden Gateway Garage",
        "DESC": "250 Clay Street",
        "INTER": "Clay between Front & Davis",
        "TEL": "(415) 433-4722",
        "OPHRS": {"OPS":            [
                    {
            "FROM": "Monday",
            "TO": "Friday",
            "BEG": "4:00 AM",
            "END": "10:00 PM"
        },
                    {
            "FROM": "Saturday",
            "BEG": "7:00 AM",
            "END": "10:00 PM"
        },
                    {
            "FROM": "Sunday",
            "BEG": "9:00 AM",
            "END": "10:00 PM"
```

```
          }
        ]},
        "OCC": "1023",
        "OPER": "1100",
        "PTS": "1",
        "LOC": "-122.3986032,37.79544154"
    },
          {
        "TYPE": "ON",
        "BFID": "650011",
        "NAME": "Sacramento St (101-199)",
        "OCC": "5",
        "OPER": "8",
        "PTS": "2",
        "LOC": "-122.3967848,37.7945224,-122.397573,37.7944219"
    },

############ 50 more Availability Records omitted#########
    ]
}
```

The data may be returned as XML (default format) and a sample XML response is shown below:

```
HTTP/1.1 200 OK
Date: Thu, 17 Feb 2011 02:06:27 GMT
Content-Length: 9570
Content-Type: text/xml; charset=utf-8
X-Powered-By: Servlet/2.5 JSP/2.1


<SFP_AVAILABILITY xmlns="http://www.sfmta.com/xsd/availability">
   <STATUS>SUCCESS</STATUS>
   <NUM_RECORDS>52</NUM_RECORDS>
   <MESSAGE>52 records found</MESSAGE>
   <AVAILABILITY_UPDATED_TIMESTAMP>2011-02-16T18:06:27.957-
08:00</AVAILABILITY_UPDATED_TIMESTAMP>
   <AVAILABILITY_REQUEST_TIMESTAMP>2011-02-16T18:06:27.910-
08:00</AVAILABILITY_REQUEST_TIMESTAMP>
   <AVL>
      <TYPE>OFF</TYPE>
      <OSPID>934</OSPID>
      <NAME>Golden Gateway Garage</NAME>
      <DESC>250 Clay Street</DESC>
      <INTER>Clay between Front &amp; Davis</INTER>
      <TEL>(415) 433-4722</TEL>
      <OPHRS>
         <OPS>
            <FROM>Monday</FROM>
            <TO>Friday</TO>
            <BEG>4:00 AM</BEG>
            <END>10:00 PM</END>
         </OPS>
```

```
        <OPS>
            <FROM>Saturday</FROM>
            <BEG>7:00 AM</BEG>
            <END>10:00 PM</END>
        </OPS>
        <OPS>
            <FROM>Sunday</FROM>
            <BEG>9:00 AM</BEG>
            <END>10:00 PM</END>
        </OPS>
    </OPHRS>
    <OCC>1021</OCC>
    <OPER>1100</OPER>
    <PTS>1</PTS>
    <LOC>-122.3986032,37.79544154</LOC>
</AVL>
<AVL>
    <TYPE>ON</TYPE>
    <BFID>650011</BFID>
    <NAME>Sacramento St (101-199)</NAME>
    <OCC>5</OCC>
    <OPER>8</OPER>
    <PTS>2</PTS>
    <LOC>-122.3967848,37.7945224,-122.397573,37.7944219</LOC>
</AVL>

############50 more Availability Records omitted#########
</SFP_AVAILABILITY>
```

If there are errors during request parameter validation or retrieving availability from
SF*park* data warehouse it returns an ERROR status and possible reason. A sample of
such an error response is shown below. Note that the HTTP status code is still 200
since the request itself was successfully processed by the server and the error is
merely data related.

```
HTTP/1.1 200 OK
Date: Thu, 17 Feb 2011 02:07:46 GMT
Content-Length: 256
Content-Type: application/json; charset=utf-8
X-Powered-By: Servlet/2.5 JSP/2.1

{
    "STATUS": "ERROR",
    "MESSAGE": "Error while retrieving availability.  Both Latitude and
Longitude are required if specifying either one of these parameters.",
    "AVAILABILITY_UPDATED_TIMESTAMP": "2011-02-16T18:07:46.999-08:00",
    "AVAILABILITY_REQUEST_TIMESTAMP": "2011-02-16T18:07:46.982-08:00"
}
```

In the above error response the error details indicate that both the request parameter LAT and LONG are required when specifying either one of these. For details about the response elements, please refer to the section SF*park* Availability Service Response.

3   SF*park* Availability Service Response

This section describes the information returned by the availability service and to describe the service response elements. The SF*park* servers are capable of sending compressed response when the request header (Accept-Encoding: gzip,deflate) is set that indicates the user can accept compressed data. Using compression allows reducing the amount of data needed to be transmitted back in the response.

3.1 *XML Response*

Availability service data response may be requested as XML or JSON. This section describes the format and elements when requesting the response to be XML (default). Note that XML response is sent if the query parameter contains response=xml or omits it (since the SF*park* default is XML) when invoking the service. It includes the following elements (only relevant elements are sent based on the outcome of the validation and retrieval of availability data from the SF*park* Database). The response XML sent back by the service is based on the SF*park* Availability XML XSD and is available at:
http://api.sfpark.org/xsd/availability.xsd

```
<STATUS>Response indicating Success or Error</STATUS>
<REQUESTID>Identifier that is returned if passed in request</REQUESTID>
<UDF1>User defined field identifier that is returned if passed in
request</UDF1>
<NUM_RECORDS>Returns the number of records being returned, if
Success</NUM_RECORDS>
<ERROR_CODE>Returns an SFMTA Error code if encountered Error</ERROR_CODE>
<MESSAGE>Returns additional details as appropriate</MESSAGE>
<VERSION>Placeholder for possible future use: To allow returning the version
of the service that created this response</VERSION>
<AVAILABILITY_UPDATED_TIMESTAMP>Returns the Timestamp of when the
availability data response was updated for the
request</AVAILABILITY_UPDATED_TIMESTAMP>
<AVAILABILITY_REQUEST_TIMESTAMP>Timestamp of when the associated request was
originally received by SFMTA that generated this response.
</AVAILABILITY_REQUEST_TIMESTAMP>
<AVL> 0 or more availability elements are passed, if SUCCESS. Each element
provides the id of the parking location along with location co-ordinates.
Currently up to two location points (Lat/Long) pairs are sent. If two pairs
```

are sent they are an approximate begin and end position of the line for the specific parking location</AVL>

The **STATUS** is the main XML element that returns SUCCESS or ERROR.

The AVAILABILITY Element has the following child elements

<TYPE>Specifies whether on or off street parking</TYPE>
<NAME>Name of parking location or street with from and to address, if available</NAME>
<DESC>Returned for OSP only — usually address for the parking location, if available</DESC>
<INTER>Returned for OSP only — usually cross street or nearby intersection from parking location, if available</INTER>
<TEL>Returned for OSP only — Contact telephone number for parking location, if available</TEL>
<OSPID>Unique SFMTA Identifier for the parking location for off street parking type</OSPID>
<BFID>Unique SFMTA Identifier for on street block face parking location</BFID>
<OCC>Number of spaces currently occupied</OCC>
<OPER>Number of spaces currently operational for this location</OPER>
<PTS>Number of location points returned for this record. Usually 1 for OSP and 2 for on-street parking location</PTS>
<LOC>Comma separated Longitude and Latitude values of points for this location. Currently a max of two points are specified for a parking location which are the begin and end points</LOC>
<OPHRS>Returned for OSP only. Returns sub elements (specified below) that indicate the operating hours schedule information for this location. It may not contain hours for special events, etc.</OPHRS>

The OPHRS Element has the following child elements.
Note: Operating hour's information is returned in 1 or more <OPS> sub-elements. If there is no schedule information in the response, it indicates that the system did not retrieve any current valid operating hour's information for this location.

<OPS>Returns an operating schedule. There may be 1 or more schedule information returned for a location.
    <FROM>Start day for this schedule, e.g., Monday</FROM>
    <TO>End day for this schedule, e.g., Friday</TO>
    <BEG>Indicates the begin time for this schedule</BEG>
    <END>Indicates the end time for this schedule</END>
</OPS>

<RATES>Returns sub elements (described below) that indicate the general pricing or rate information for this location. Note that the rate information provided for on-street location is updated daily (soon after midnight) and returns only the rates that apply for the day of the request. It may not contain rates for special event, motorcycle rates, etc.</RATES>

The RATES Element has the following child elements.
Note: Rate information is returned in 1 or more <RS> sub-elements. If there is no RATE information found for a location the response will return a RATE of 0 with a rate qualifier (RQ) message similar to "See Meter" or "See Garage" for off street garage locations.

```
<RS>Returns a rate schedule. There may be 1 or more rate schedules for a
location.
     <BEG>Indicates the begin time for this rate schedule</BEG>
     <END>Indicates the end time for this rate schedule</END>
     <RATE>Applicable rate for this rate schedule</RATE>
     <DESC>Used for descriptive rate information when not possible to
specify using BEG or END times for this rate schedule</DESC>
     <RQ>Rate qualifier for this rate schedule, e.g. Per Hr</RQ>
     <RR>Rate restriction for this rate schedule if any</RR>
</RS>
```

Note: The percentage of parking spaces available for a location may be calculated using the values returned. E.g., `% availability for a location = ( (OPER – OCC) / OPER ) * 100 %`
It is recommended to verify that oper is non-zero to avoid mathematical division by 0 error.

Also the format for the location is comma separated paired values in the conventional order (also SF*park* standard unless otherwise noted) of Longitude, Latitude. Returned values should be paired accordingly to identify unique points. For a single pair of values returned it indicates a specific geographical point. For two points, it represents the two end points of an approximate line that indicates the parking location. Currently, max of two points (or four comma separated values) is returned for any location. However, SF*park* may in future switch to returning multiple point data which may need to be handled differently.

Note that if incomplete parameters or arbitrary data is sent when invoking this service, the error response may not be available.

3.1.1 Successful Response Format

```
<STATUS>SUCCESS</STATUS>
<AVAILABILITY_UPDATED_TIMESTAMP>2011-02-18T13:17:13.000-
08:00</AVAILABILITY_UPDATED_TIMESTAMP>
```

Note: SUCCESS in the STATUS is sent and indicates that the input request parameters were valid. This should provide 0 or more availability records from the

SF*park* database based on the request parameters. Also, the field AVAILABILITY_UPDATED_TIMESTAMP indicates the time when the response was queried and calculated by the data warehouse. It applies to the entire set of records returned in the response. Individual availability records are a snapshot of the status in the data warehouse at the time this response is queried based on the individual events that were received by the SF*park* system. The actual status may have changed since the events are transmitted from the individual spaces and stored and processed prior to returning this data.

The SF*park* processing includes request search parameter validation followed by retrieval of the availability data from the SF*park* Data Warehouse based on the request parameters.

Below are the response examples showing the phase which encountered the error.

3.1.2 Parameter Validation Error Response Format

```
<STATUS>ERROR</STATUS>
<MESSAGE>Error while retrieving availability. Search parameters are not
valid. long should be numeric
</MESSAGE>
```

ERROR message above indicates error is during the parameter validation phase of the request which contains an invalid search request element LONG (longitude). It indicates that the element must be a valid number.

3.1.3 Retrieving from Database Error Response Format

```
<STATUS>ERROR</STATUS>
<MESSAGE>Error while retrieving Availability</MESSAGE>
```

ERROR message above indicates error is while retrieving the data from the database.

Please review the ERROR elements in the response for information as to the cause of the error and whether it failed during request parameter validation or retrieval from SF*park* database. These may include technical error details which may need to be reviewed for the specific cause. The error should provide information to identify why the invocation failed. If the cause is request parameter related, please fix the problems for avoiding similar errors in the future.

If the error is during the database phase it is assumed that the request parameter validation was successful. For this error, the response will need to be reviewed for

determining if the cause is due to specific request value sent (e.g., latitude or longitude values not in proper range) or other issues entirely related to SF*park* system.

If such an error occurs, please try the request again later. If the problem persists, you may submit the error details to SF*park* if unable to confirm the cause.

3.2 *JSON Response*

Availability service response may be requested as XML or JSON. This section describes the elements sent when requesting the response as JSON. Note that service response is sent as JSON whenever the query parameter contains response=json when invoking the service, otherwise the response will be sent as XML (default).

The JSON response sent back by the service contains elements in JSON format and are according to the SF*park* Availability XML XSD. For reference, the latest version of the XSD is available at: http://api.sfpark.org/xsd/availability.xsd

The response includes the following elements (only relevant elements are sent based on the outcome of the validation and retrieval of availability data from the SF*park* Database).

```
{
    "STATUS": "Response indicating SUCCESS or ERROR ",
    "REQUESTID": "Identifier that is returned if passed in request ",
    "UDF1": "User defined field identifier that is returned if passed in
request",
    "NUM_RECORDS": "Returns the number of records being returned, if Success
",
  "ERROR_CODE ": "Returns a SFMTA internal Error code if encountered Error ",
    "MESSAGE": "Returns additional details as appropriate ",
    "VERSION": "Placeholder for possible future use: To allow returning the
version of the service that created this response",
    "AVAILABILITY_UPDATED_TIMESTAMP": "Returns the Timestamp of when the
availability data response was updated for the request",
    "AVAILABILITY_REQUEST_TIMESTAMP": "Timestamp of when the associated
request was originally received by SFMTA that generated this response.",
    "AVL":    [0 or more availability elements are passed, if SUCCESS.
Availability elements are described below.
           {
        "TYPE": "Specifies whether on or off street parking",
        "NAME": "Name of parking location or street with from and to
address, if available",
        "DESC": "Returned for OSP only — usually address for the parking
location, if available",
        "INTER": "Returned for OSP only — usually cross street intersection
parking location, if available",
```

            "**TEL**": "Returned for OSP only — Contact telephone number for parking
location, if available",
            "**OSPID**": "Unique SFMTA Identifier for the parking location for off
street parking type",
            "**BFID**": "Unique SFMTA Identifier for on street block face parking ",

            "**OCC**": "Number of spaces currently occupied ",
            "**OPER** ": "Number of spaces currently operational for this location
",
            "**PTS**": "Number of location points returned for this record. Usually
1 for OSP and 2 for on-street parking location ",
            "**LOC**": "Comma separated Longitude and Latitude values of points for
this location. Currently a max of two points are specified for a parking
location which being the start and end points",
            "**OPHRS**": "Returned for OSP only. Returns array of sub elements
(specified below) that indicate the operating hours schedule information for
this location. It may not contain hours for special events, etc."
            "**RATES**": "Returns array of sub elements (described below) that
indicate the general pricing or rate information for this location. Note that
the rate information provided for on-street location is updated daily (after
midnight) to apply for the day of the request. It may not contain rates for
special event, motorcycle rates, etc."
        }
    ]
}

The **OPHRS** Element indicated above has the following child elements.
Note: Operating hour's information is returned as an array of one or more OPS
sub-elements. If there is no schedule information in the response, it
indicates that the system did not retrieve any current valid operating hour's
information for this location.

"**OPHRS**": {"**OPS**":   "Returns an operating schedule. There may be 1 or more
schedule information returned for a location"
 [
                {
                    "**FROM**": "Start day for this schedule, e.g., Monday",
                    "**TO**": "End day for this schedule, e.g., Friday",
                    "**BEG**": "Indicates the begin time for this schedule,
e.g., 4:00 AM",
                    "**END**": "Indicates the end time for this schedule, e.g.,
10:00 PM"
            }
]

The **RATES** element indicated above has the following child elements.
Note: **RATES** information is returned as an array of one or more **RS** or rate
schedule elements.
If there is no RATE information found for a location the response will return
a RATE of 0 with a rate qualifier (RQ) message indicating "See Meter" or "See
Garage" for off street garage locations.

```
"RATES": {"RS":  "Returns a rate schedule. There may be 1 or more rate
schedules for a location"
[
                {
                        "BEG": " Indicates the begin time for this rate
schedule",
                        "END": "Indicates the end time for this rate schedule ",
                        "RATE": "Applicable rate for this rate schedule",
                        "DESC": "Used for descriptive rate information when not
possible to specify using BEG or END times for this rate schedule",
                        "RQ": "Rate qualifier for this rate schedule, e.g. Per
Hr",
                        "RR": "Rate restriction for this rate schedule, if any"
                },
]
```

The **STATUS** is the main XML element that returns SUCCESS or ERROR.

Note: The percentage of parking spaces available for a location returned may be calculated as: `((OPER_SPACE_CNT – OCCUPIED_SPACE_CNT)/OPER_SPACE_CNT) * 100 %` It is recommended to verify that oper is non-zero to avoid mathematical division by 0 error.

The format for the location is comma separated paired values in the conventional order (also SF*park* standard unless otherwise noted) of Longitude, Latitude. Returned values should be paired accordingly to identify unique points. For a single pair of values returned it indicates a specific geographical point. For two points, it represents the two end points of an approximate line that indicates the parking location. Currently, max of two points (or four comma separated values) is returned for any location.

Note that if incomplete parameters or arbitrary data is sent when invoking this service, the error response may not be available.

3.2.1 Successful Response Format
```
"STATUS": "SUCCESS"
"AVAILABILITY_UPDATED_TIMESTAMP": "2011-04-18T13:17:13.000-07:00"
```

Note: SUCCESS in the STATUS is sent and indicates that the input request parameters were valid. This should provide 0 or more availability records from the SF*park* database based on the request parameters.

The SF*park* processing includes request search parameter validation followed by retrieval of the availability data from the SF*park* Data Warehouse based on the

parameters specified in the request. Below are the response examples showing the phase which encountered the error.

3.2.2 Parameter Validation Error Response Format

```
"STATUS": "ERROR"
"MESSAGE": "Error while retrieving availability. Both Latitude and
Longitude are required if specifying either one of these parameters."
```

The MESSAGE element may provide further details as shown above. ERROR message above indicates error is during the parameter validation phase of the request and indicates that both Lat and Long need to be specified if any one of these passed.

3.2.3 Retrieving from Database Error Response Format

```
"STATUS": "ERROR"
"MESSAGE": "ERROR while retrieving Availability"
```

ERROR message above indicates error is while retrieving the data from the database.

Please review the ERROR elements in the response for information as to the cause of the error and whether it failed during request parameter validation or retrieval from SF*park* database. These may include technical error details which may need to be reviewed for the specific cause. The error should provide information to identify why the invocation failed. If the cause is request parameter related, please fix the problems for avoiding similar errors in the future.

If the error is during the database phase it is assumed that the request parameter validation was successful. For this error, the response will need to be reviewed for determining if the cause is due to specific request value sent (e.g., latitude or longitude values not in proper range) or other issues entirely related to SF*park* system. If such an error occurs, please try the request again later. If the problem persists, you may submit to SF*park* and provide details along with the error response if you are unable to confirm the cause.

4 Pricing Information

As discussed in earlier sections, the availability service allows for returning the pricing (or rates) information along with the availability data and is returned only if pricing has been specifically requested as a parameter to the service. This allows limiting the amount of data that is returned when pricing information is not required. The rates information for on street metered locations is usually structured differently than those for garages. On street metered locations usually have a rate associated with various time periods and may change during the day. For both metered on-street and garages or off-street locations multiple rates may apply for a location based on time, etc. Hence the returned data may contain more than one rate information for a location. If there is no RATE information found for a location the response will return a RATE of 0 with a rate qualifier (RQ) message similar to "See Meter" or "See Garage" for off street garage locations.

This section will describe the formats of data that may be returned and how it can be classified into two types for purposes of presenting the information on a display.

4.1 *General Guidelines*

The following are some guidelines when interpreting the rate information returned by the service.

- All pricing data is in US Dollars.
- On street parking locations (block-face) pricing is mostly structured by BEG and END time along with the applicable RATE for that period. A rate of 0 indicates either no-charge or in certain situations some other reason that the rate is not returned, e.g., the information is not available to the service. A rate qualifier is always passed which will usually specify whether it is no-charge. If no rate was found for a location then the rate qualifier indicates "See Meter". Hence, when the rate is returned as 0, just use the description in the rate qualifier as that will describe the rate more appropriately.
- If there is a BEG and END time being returned then that rate schedule does not require a DESC and hence it will not be returned. The DESC allows to return special types of rates that are not simple to be returned as a BEG or END time, e.g., "Early Bird". Hence these may be considered as mutually exclusive types. A DESC is returned along with some Rate Qualifier, e.g., "Flat Rate". A Rate Restriction may be returned as well, e.g., "Sat: In after 7AM/Out by 4AM;Sun: In

after 4AM/Out by 4AM". The DESC and associated elements allow returning special pricing that usually apply to garages only and not metered locations.

- Certain off street locations are metered lot and in those situations the rates returned for these will be similar to those returned for on street locations.
- In certain cases, there is a need for longer text to be returned for rate restrictions. Ex: "Sat: In after 7AM/Out by 4AM;Sun: In after 4AM/Out by 4AM." In the example the text is readable as-is, however to improve readability on a limited screen size (e.g., mobile devices) the semi-colon is used as a special char within the text to be a hint to allow replacing it with a line-break. This will allow generically replacing the ';' with a line-break when appropriate for display purposes.
- Off street locations may return combination of BEG-END data and DESC type rate schedule records as appropriate. For on-street locations it is expected that only BEG-END data elements will be returned.

4.2 *Examples of rate information*

The following are some examples to help illustrate displaying the rate information returned from the service. A rates array contains one or more rate schedule elements. However, it can be assumed that a particular rate schedule element will be either BEG and END or DESC type of rate data. Hence, there are essentially only two types of display formats to be considered, either a "BEG-END" (time interval) data or "DESC" (descriptive) rate data as discussed below.

4.2.1 BEG and END data example

Following is an example of RATES array consisting of multiple rate schedules BEG and END data elements. To keep it simple the example does not include DESC elements.

```
<RATES>
        <RS>
           <BEG>12:00 AM</BEG>
           <END>7:00 AM</END>
           <RATE>0</RATE>
           <RQ>No Charge</RQ>
        </RS>
        <RS>
           <BEG>7:00 AM</BEG>
           <END>6:00 PM</END>
           <RATE>3.5</RATE>
           <RQ>Per Hour</RQ>
        </RS>
```

```
<RS>
    <BEG>6:00 PM</BEG>
    <END>12:00 AM</END>
    <RATE>0</RATE>
    <RQ>No Charge</RQ>
</RS>
</RATES>
```

Using above as an example a simple display format is shown below: For actual display, special columns or other appropriate characters may be used as appropriate to provide better readability.

Note that rate qualifier (RQ) is expected to be returned with each element as shown in above example where Rate of '0' has a rate qualifier of "No Charge". For Rate element of '3.5' it is being displayed as '$3.50' as appropriate currency format. The BEG and END times are concatenated with a '-' char to signify the time duration the rate is applicable.

12:00 AM – 7:00 AM	No Charge
7:00 AM – 6:00 PM	$3.50 Per Hour
6:00 PM – 12:00 AM	No Charge

4.2.2 DESC data example

If the rate schedule contains DESC element then presence of this is used as an indicator of descriptive type for rate display which is used when the rate cannot be classified as a time interval. Following is an example of a RATES element array consisting of rate schedules with only DESC data elements. To keep it simple the example does not include BEG or END element data.

```
<RATES>
    <RS>
        <DESC>Incremental</DESC>
        <RATE>3.5</RATE>
        <RQ>Per 1/2-Hr</RQ>
    </RS>
    <RS>
        <DESC>24-Hour Max/Lost Tkt</DESC>
        <RATE>36</RATE>
        <RQ>Flat Rate</RQ>
    </RS>
    <RS>
        <DESC>Early Bird</DESC>
        <RATE>20</RATE>
        <RQ>Flat Rate</RQ>
```

```
            <RR>Mon-Fri: In by 10AM/Out by 7PM</RR>
        </RS>
        <RS>
            <DESC>Weekend</DESC>
            <RATE>7</RATE>
            <RQ>Per Day</RQ>
            <RR>Sat: In after 7AM/Out by 4AM;Sun: In after 4AM/Out by
4AM</RR>
        </RS>
</RATES>
```

Using above as an example a simple display format is shown below: For actual display, special columns or other appropriate characters may be used as appropriate to provide better readability.

Note that rate qualifier (RQ) is expected to be returned with each element. Rates are formatted with appropriate $ symbol and decimal, e.g., rate of '36' is displayed as '$36.00'. The rate restriction is added, if available as it may not be returned in all cases. This allows specifying any restriction to the rate as applicable.

Incremental Per ½-Hr	$3.50
24-Hour Max/Lost Tkt Flat Rate	$36.00
Early Bird Rate *Mon-Fri: In by 10AM/Out by 7PM*	$20.00 Flat
Weekend *Sat: In after 7AM/Out by 4AM* *Sun: In after 4AM/Out by 4AM*	$7.00 Per Day

4.2.3 Combination data example

If the rates contain mixed rate schedule data including both BEG-END and DESC elements then the rate display should use a combination of the approaches discussed for the two individual formats. Since each rate schedule can be displayed as one row of data on a display you can mix and match as per the data that gets returned. A sample table is shown below that contains a mix of these two types of data.

6:00 AM – 7:00 PM	$3.50 Per Hour
24-Hour Max/Lost Tkt	$36.00 Flat Rate

Early Bird	$20.00 Flat Rate
Mon-Fri: In by 10AM/Out by 7PM	
Evening/Weekend	$7.00 Per Day
Sat: In after 7AM/Out by 4AM	
Sun: In after 4AM/Out by 4AM	
Evenings: In after 7:00 PM/Out by 6:00 AM	

The above examples are for illustrative purposes only and may not represent actual data that may be returned. However, it helps illustrate how to classify it into two formats for displaying actual data that will be returned. It should be noted that the rate information and presentation is subject to change so certain exceptions or alternate format may be needed in future as needed.

5 Sample Error Messages

This section provides a few sample error messages returned when invoking the service. It is intended as a concise supplement to the prior section "SF*park* Availability Service Response" and may include information already discussed in that section. The most common errors to expect are during the validation of the search parameters passed to the service. This section is thereby focused on providing samples from such validation errors that may occur.

5.1 *Validation Errors*

The following samples illustrate the validation error responses for REST service invocations.

The status will always be ERROR as shown below. The ERROR status indicates that there was problem during validation or retrieving availability data for this request. Hence if you do receive this status then review the error message to understand the reason of the problem.

Error Status for XML Response: <STATUS>ERROR</STATUS>

Error Status for JSON Response: STATUS: "ERROR"

Following are some sample error messages that may be encountered during an invocation of the service. Only relevant MESSAGE element is shown for these sample validation error scenarios:

Scenario 1: Numeric Field contains alpha character

MESSAGE: "Error while retrieving availability. **Search parameters are not valid. long should be numeric.**"

Scenario 2: Field contains invalid option for this type

MESSAGE: "Error while retrieving availability**. Search parameters are not valid. long should be numeric. Allowed values for uom are: mile, km, foot, meter, m, yard.**"

Scenario 3: Request contains invalid element. Here LATITUDE is not a valid element, it should be LAT and hence the error

MESSAGE: "Error while retrieving availability. **Search parameters are not valid. latitude is not a valid request parameter."**

Scenario 4: General Error Response

MESSAGE: "Invalid parameters found in request. **Valid parameters are requestId, lat and long (both or none), radius, uom (mile, km, foot, meter, m, yard), response (xml or json), jsoncallback (callback method name for jsonp support), type (on, off or all), version, pricing (yes, no), udf1 and method (availability).**

Scenario 5: JSONP invalid method name Error Response

MESSAGE: "**Error while retrieving availability. JSON callback parameter is not valid. It should be non-empty string of less than 100 char and may not start with jsonp**."

Scenario 6: JSON not specified as response type when requesting JSONCALLBACK

Error while retrieving availability. Response type needs to be specified as JSON if JSON callback is requested.

Scenario 7: More than one validation error found

The error response usually tries to parse all errors and report them all in case of multiple validation errors. An example is shown below which contained two validation errors.

MESSAGE: "Error while retrieving availability. **Search parameters are not valid. radius should be numeric. Allowed values for uom are: mile, km, foot, meter, m, yard**."

5.2 *Retrieving from Database Error Response Format*

The general error message format for database errors is provided below. To understand the reason of the error it may be needed to review the message to

understand the reason of the problem. In certain cases there may not be any further reason, in which case you may want to retry the request later. This may happen during SF*park* database being unavailable, etc. These conditions may also happen during scheduled maintenance periods. If such errors are encountered frequently and no details are present in the message and they continue after repeated attempts to differing request parameters, you may submit these details to SFpark.

For failed invocations the status will always be ERROR as shown below. The ERROR status indicates that there was some problem during retrieving availability data for this request. The message will indicate that the error was during retrieving availability. This indicates that the request was valid and that the error happened during retrieving from database.

> "STATUS": "ERROR"
> "MESSAGE": "ERROR while retrieving Availability"

ERROR message above indicates error is while retrieving the data from the database.

5.2.1 No Records Found Response Format

It is possible that a request query returns no availability records (e.g., long/lat in the search is a point outside the SF*park* area) in which case the response will indicate that there are 0 records found. This is normal for such requests as there is no valid parking location found to report availability for the request and is not an ERROR. A sample response is shown below.

```
<STATUS>SUCCESS</STATUS>
<NUM_RECORDS>0</NUM_RECORDS>
<MESSAGE>0 records found</MESSAGE>
```

6 Availability Search Parameters

This section lists the availability service search parameters and their descriptions along with their default values, if any. Note that the parameters and their default values are subject to change. The SF*park* Availability service returns data matching the search criteria as specified when invoking the service. The service will use default values for certain parameters when not specified. However, if specifying a value for certain parameters it may be necessary to provide a suitable value for its complementary parameter, e.g., Latitude and Longitude.

These are specified as query parameters. The parameters are not case sensitive for the REST service but they should be valid as far as their naming and values. For further information on sample request and response message please refer to the response section.

6.1 *Availability Service Input Request Parameters*

The following request parameters are currently supported by the Availability Service. Note that all parameters are currently optional. If no parameter is supplied, then the service uses internal defaults for LAT, LONG, RADIUS and UOM.

6.1.1 REQUESTID – Request Identifier

This optional request parameter allows correlating a response to a particular request or may be used for tracking purposes. If passed, this identifier is returned as is in the response generated for the particular request. It has no other purpose in determining the outcome of the request.

Example: REQ1234567890
Default value (if none specified): Optional field – no default provided
Allowed values: Any alphanumeric string. Note: Certain special characters may not be supported and is restricted to less than 100 chars.

6.1.2 LAT – Latitude in decimal format to be specified along with LONG

This request parameter is used in conjunction with the specified Longitude parameter and represents the geographical point from which the search results will be centered.

Note: System will use a default value if both LAT and LONG are skipped. If either parameter is passed then both LAT and LONG are required.

Example: 37.7853
Default value (if both LAT and LONG are skipped): 37.7819

6.1.3   LONG – Longitude in decimal format to be specified along with LAT

This request parameter is used in conjunction with the specified Latitude parameter and represents the geographical point from which the search results will be centered.

Note: System will use a default value if both LAT and LONG are skipped. If either parameter is passed then both LAT and LONG are required.

Example: -122.4077
Default value (if both LAT and LONG are skipped): -122.4200

6.1.4   RADIUS – Search Radius

This request parameter is used in conjunction with the specified UOM parameter and represents the search radius the result will return from the requested location point. If UOM is not passed, then the service will use the default value for UOM.

Note: If no UOM is specified but RADIUS is specified, then UOM is still defaulted to mile. So be aware of these default values and their behavior and hence it is recommended to specify both RADIUS and UOM or leave them out to use the SF*park* default, currently 0.25 mile radius.

Example: 0.5
Default value (if none specified):0.25

6.1.5   UOM – Unit of Measurement for the search radius

This request parameter is used in conjunction with the specified RADIUS parameter and represents the unit of measurement for the radius parameter. The result will return

available data points based on the requested radius in this unit of measurement from the requested location point. If RADIUS is not passed, then the service will use the default value for RADIUS.

Note: If no RADIUS is specified but UOM is specified as other than mile, then radius is still defaulted to 0.25. So be aware of these default values and their behavior and hence it is recommended to specify both RADIUS and UOM or leave them out to use the SF*park* default, currently 0.25 mile radius.

Example: mile
Default value (if none specified): mile
Allowed values: mile, km, foot, meter, m, yard

6.1.6   RESPONSE – Format of Response Data

This request parameter is used to specify the format of the data to be returned. Default value is XML format. However, JSON format may be specified if prefer to have data in JSON format. Note that if using the JSON callback option then the response needs to be specified as JSON.

Example: json
Default value (if none specified): xml
Allowed values: xml, json

6.1.7   JSONCALLBACK – Request Identifier

This request parameter allows support for JSONP or JSON with padding. JSONP is script tag injection, passing the response from the server to a user specified function. The JSONCALLBACK parameter allows specifying the function name to use in the response. If specified, the JSON response is wrapped in the requested call back, e.g., "mycallback(<JSON AVAILABILITY REPONSE>)". It requires that the response be specified to be returned as JSON and is not supported for XML response data. It has no other purpose in determining the outcome of the request.

Example: mycallback
Default value (if none specified): Optional field – no default provided
Allowed values: Any alphanumeric string. Note: Certain special characters may not be supported.

Special Rules: Response type needs to be json. The method name may not be empty string. It also cannot start with jsonp and is restricted to less than 100 chars.

6.1.8 TYPE – Limit data to specified parking type

This request parameter is used to specify the data returned be limited to the requested parking type. There are currently following two parking types supported, on-street (on) and off-street (off). Use the parameter to allow restricting data to following parking types; on-street (on), off-street (off) or all (returns both on and off-street or all results).

Example: all
Default value (if none specified): all
Allowed values: all, on, off

6.1.9 METHOD – Method to invoke

This request parameter is used invoke available methods for the availability service. Currently only the availability method is provided. In future other methods may be supported.

Example: availability
Default value (if none specified): availability
Allowed values: availability

6.1.10 PRICING – Include rate information in response

This request parameter is used to specify whether the data returned should include the pricing information for the parking locations included in the response. The rate information does not change frequently but involves extra information to be added to the response. Hence, requests to retrieve pricing data should be made specifically by setting the pricing option. The rate information returned by the service is discussed in an earlier section.

Example: yes
Default value (if none specified): no
Allowed values: no, yes

6.1.11 UDF1 – User Defined Field # 1

This optional request parameter allows user to pass in a text string to be used as a field that they may define for their internal use, e.g., specifying an organization name or may be used for tracking purposes. If passed, this identifier is returned as is in the response generated for the particular request. It has no other purpose in determining the outcome of the request.

Example: "SFMTA"
Default value (if none specified): Optional field – no default provided
Allowed values: Any alphanumeric string. Note: Certain special characters may not be supported and is restricted to less than 100 chars.

6.1.12 VERSION – Placeholder for possible future use

Note that this is currently a placeholder for possible future use and is ignored at this time. However, if specifying this request parameter it should be set to no more than 100 chars. It currently has no other purpose in determining the outcome of the request.

Example: "1.0"
Default value (if none specified): Optional field – no default provided
Allowed values: Any alphanumeric string less than 100 chars. Note: In future it may be restricted to specific valid version strings as appropriate.

7 Using soapUI for testing the REST based Service

SoapUI is a powerful open source web service testing tool and is available from http://www.soapui.org. SF*park* doesn't make any recommendations on testing strategy or tools and this section is provided merely as an aid in setting up testing for the availability service.

SoapUI has capability to test REST Services which can optionally be tested using a regular browser; however, they have limited capability so testing using SoapUI is shown below that will provide a good test environment. This section provides basic steps to assist with setting up the test tool and how to invoke the REST based service for testing. The samples provided below are based on SoapUI version 3.5. Newer versions may look slightly different but main steps should be similar. Alternate testing tools or means may also be used as suitable for your environment as appropriate.
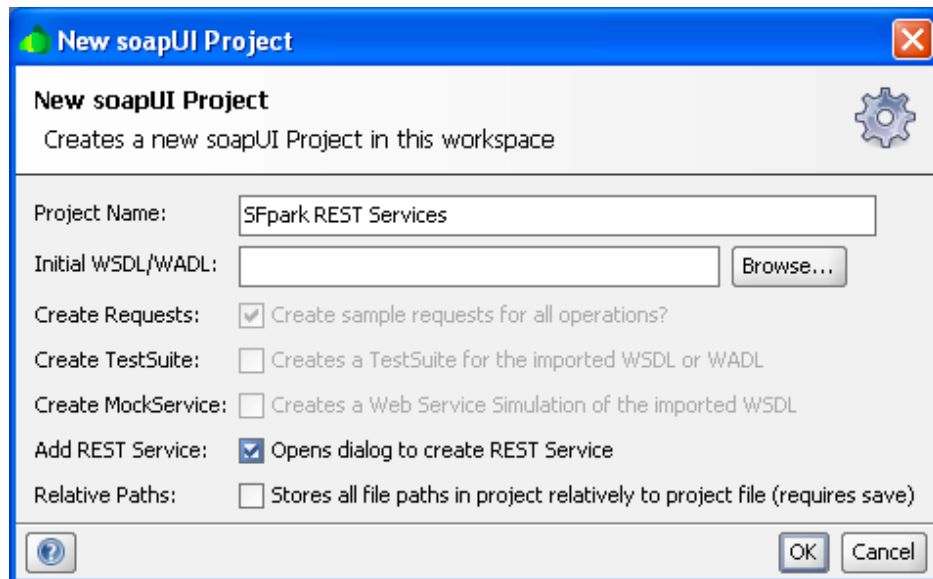
Launch the soapUI console and click on the icon to create a new project (or select from File option menu to create a new project)
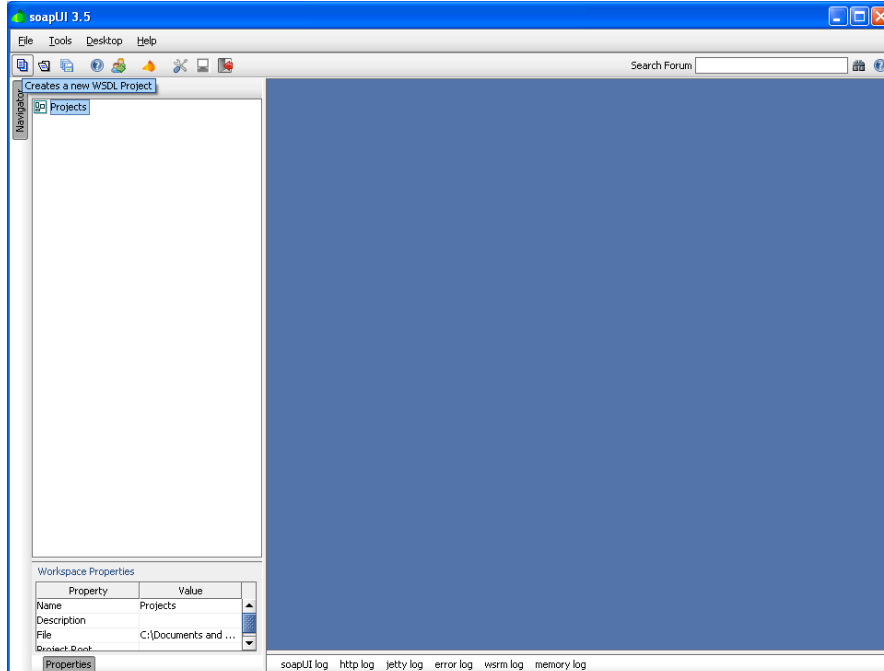
Next Enter a project name and select the option "Opens dialog to create REST service"
    Project Name: SF*park* REST Services
    Check Option: Opens dialog to create REST service
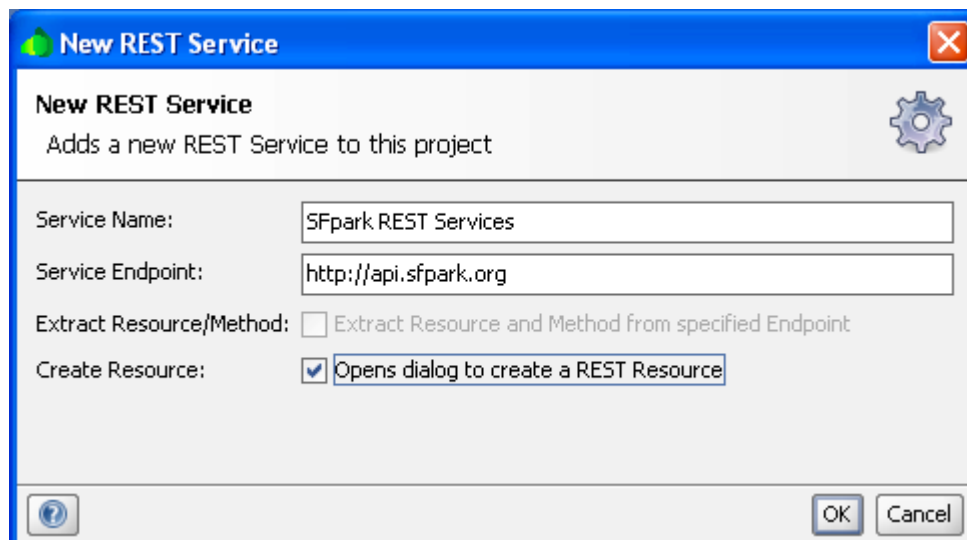
This will pop up a new window to enter a new REST Service. Provide a suitable Service Name and the Service Endpoint (need to enter only the server IP/name and port number).

NOTE: The screen below shows the current SF*park* API service DNS name (port number is not required as the service uses the HTTP default port 80).

Service Name: SF*park* REST Services
Service Endpoint: http://api.sfpark.org
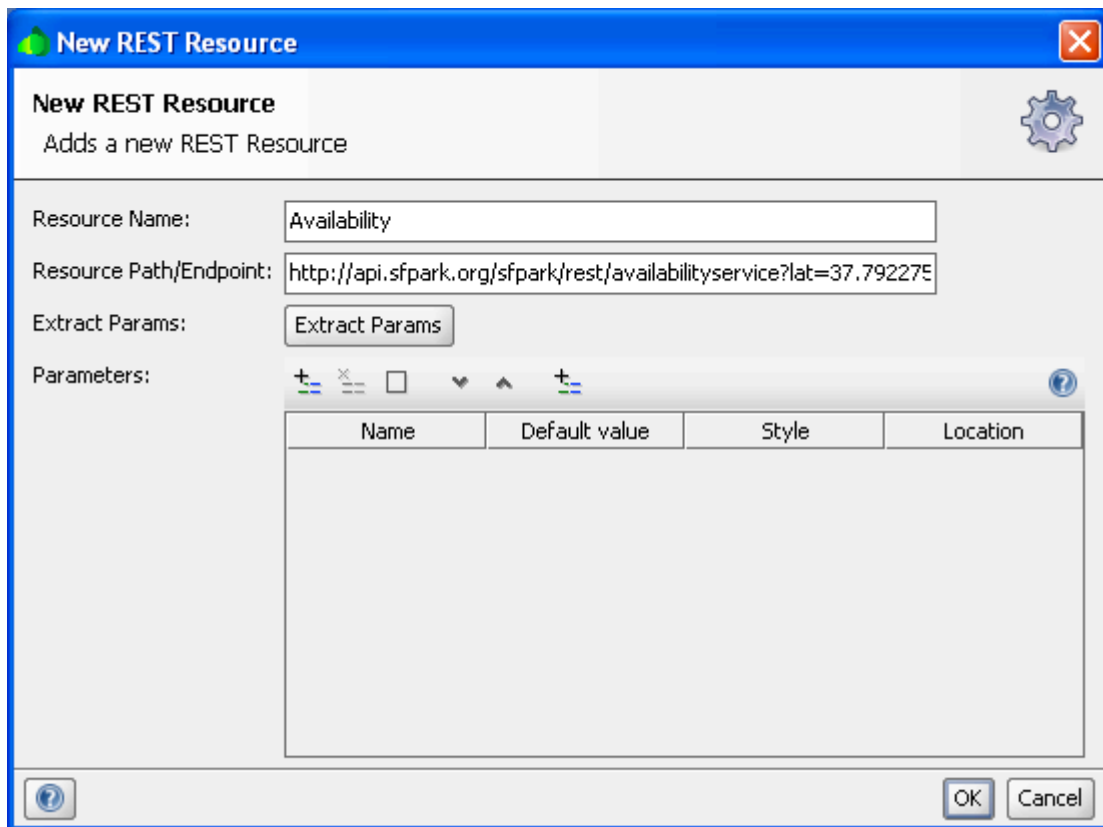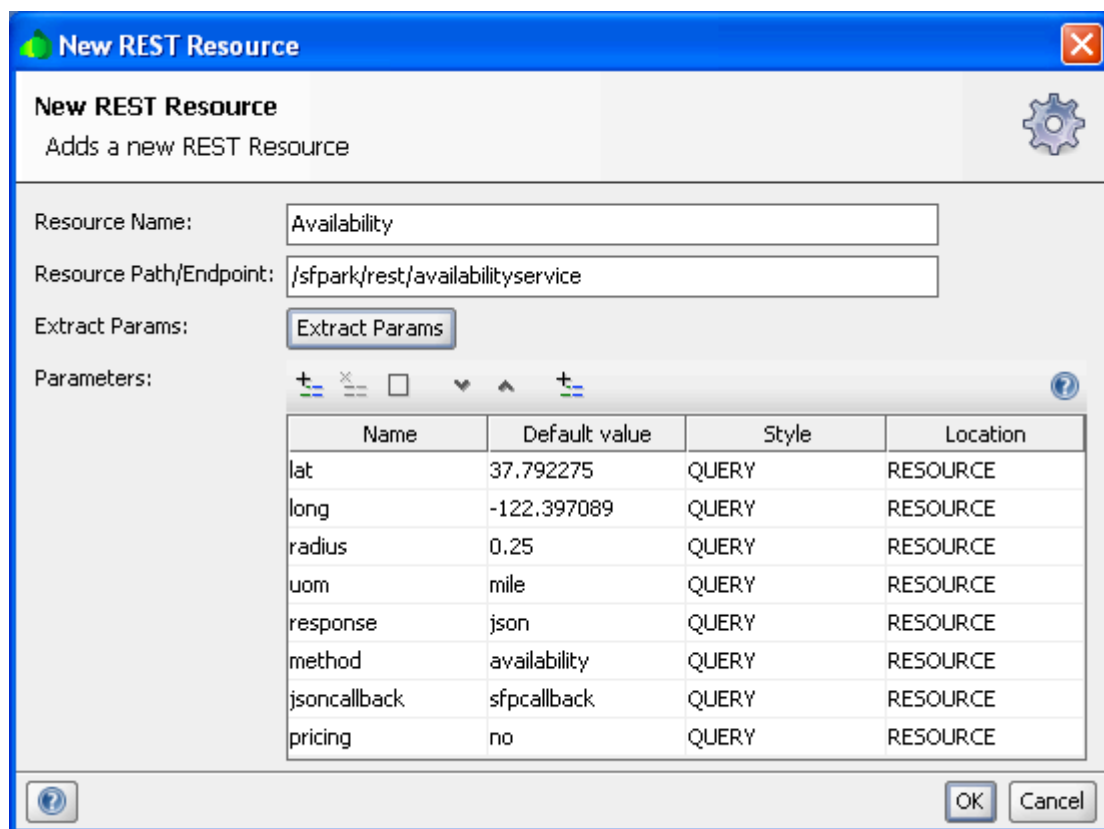
Clicking OK should pop up a new window to enter a new REST Resource. Provide a suitable Resource Name and the Resource Path/ Endpoint as shown below.

For the Resource Path type in similar to below: This includes some of the important parameters for the service along with some sample values. Update it as appropriate for your tests.

http://api.sfpark.org/sfpark/rest/availabilityservice?lat=37.792275&long=-122.397089&radius=0.25&uom=mile&response=json&method=availability&jsoncallback=sfpcallback&pricing=no

Next: In the above screen you can then simply click on the "**Extract Params**" button and the tool will review the URL and set up all the parameters for this service as shown below:

NOTE: The Resource Path gets truncated after the Parameters have been extracted. You can also update the default value for each parameter shown in the above screen by left clicking in the value for a particular parameter. Leave Style selection and Location to default of QUERY and RESOURCE.

Optionally you could enter the parameters manually by clicking the "+" icon in the Parameters section to add the parameters allowed for this service.

Next click "OK" button. This will launch the New REST Method dialog screen as shown.
Enter a suitable Method Name, e.g., Availability.
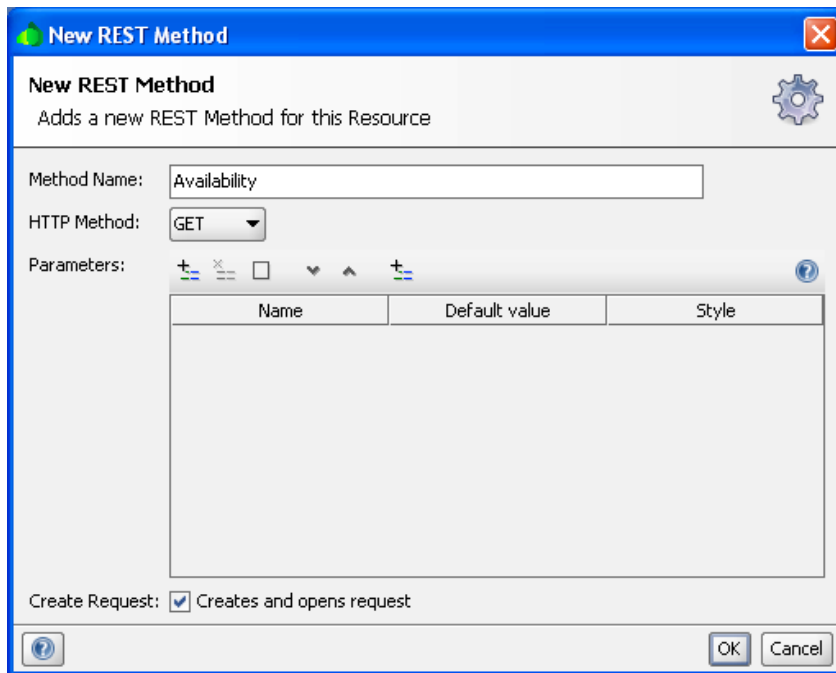
Leave the other selections to default and click OK to launch the Test window. The test window is shown below (shown maximized).
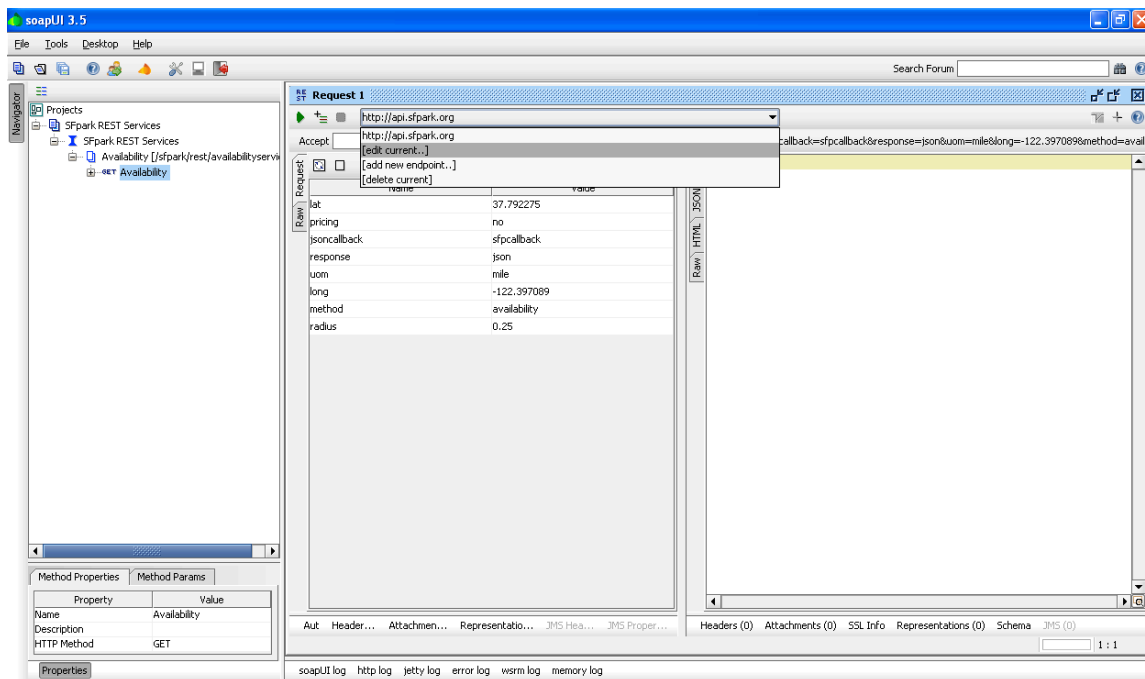
You can now start testing the service by clicking the green Play Icon.
NOTE: If you need to change the SF*park* server endpoint you can simply edit the URL at the top as shown in the above screen. Left click on the URL as highlighted and select the option "edit current". Then replace the server IP address to the desired server name or IP address.

Once you submit the request (by clicking the green play icon) the server should respond with a response message along with relevant information. To view the raw response that includes the HTTP headers, click the Raw Tab to the rightmost pane.

If the response is taking too long (> 30seconds) there could be connectivity problems as the server may be down or not reachable. Also, if you are testing from a corporate network ensure the firewall rules allow connecting to the SF*park* server.

If you receive a SUCCESS response similar to below it has successfully returned the availability information for the requested search parameters. Note: since we requested jsoncallback in the above request select the "Raw" tab in the right pane to view the response as the Content-Type is set to application/javascript for JSON callback (JSONP) option.
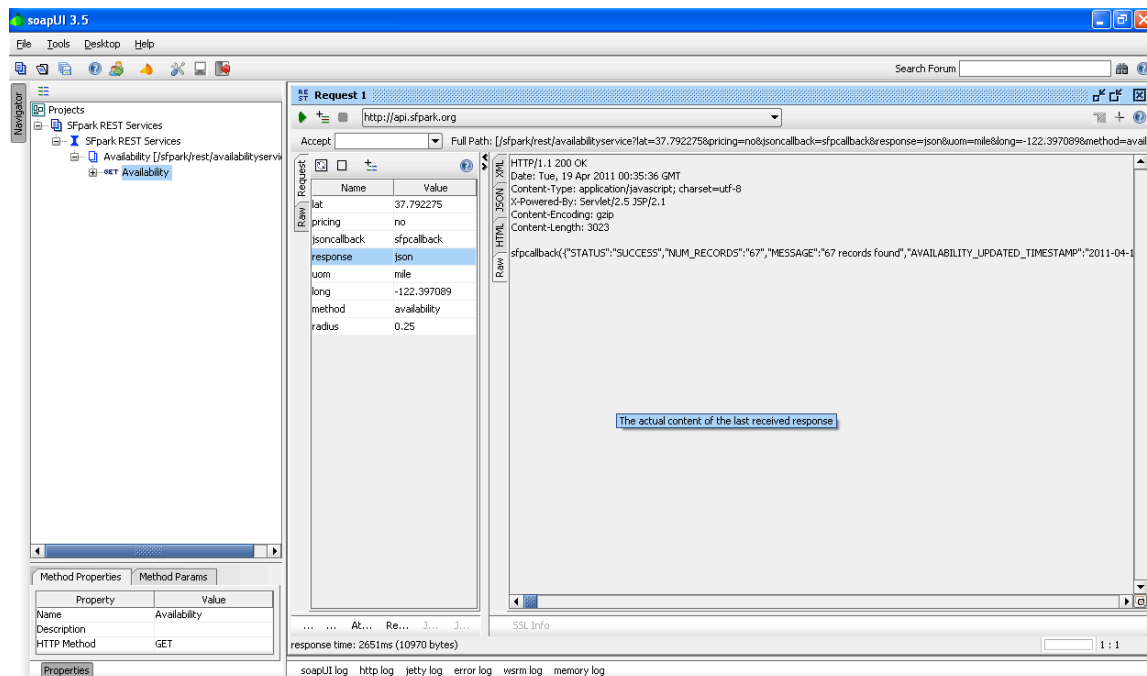
Looking at the data returned the STATUS for the request resulted in SUCCESS and NUM_RECORDS field indicates the actual number of results found matching the request criteria.

If the JSON callback is not needed for the response, clear out the value for the jsoncallback parameter and re-submit the request to receive response in JSON format as shown below.

NOTE: If requesting jsoncallback option then the response type needs to be specified as json. This option is not valid when response type of XML is specified.
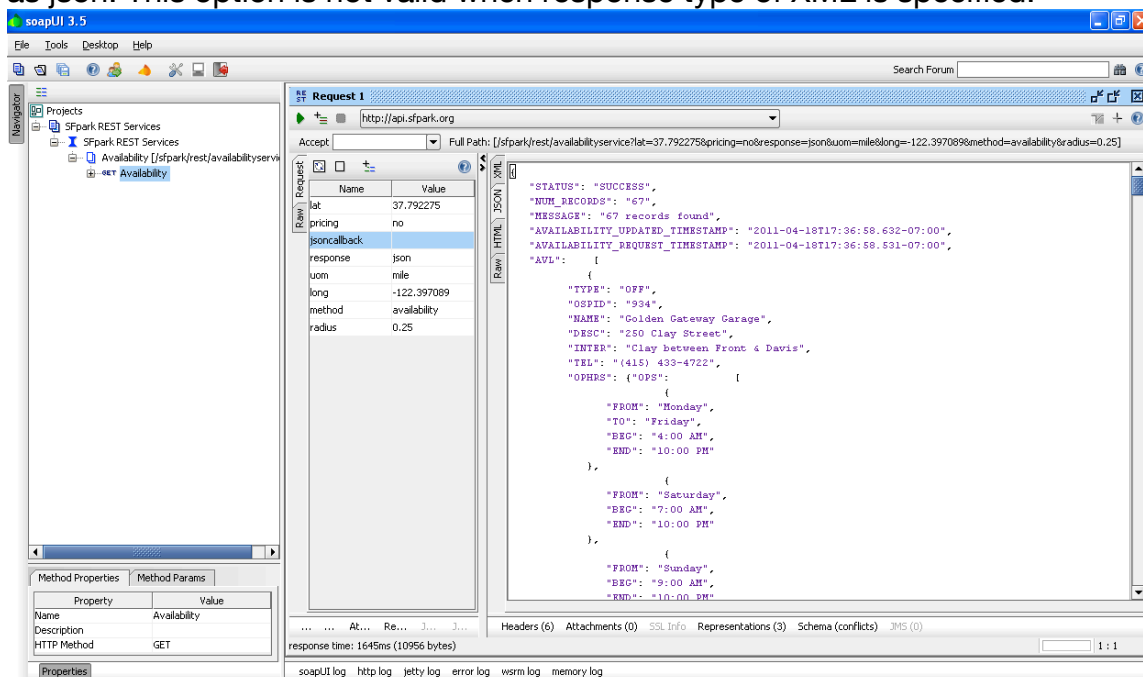
In this case the response Content-Type is now set to application-json and HTTP status code: 200.
Tip: This is seen by clicking the "**Raw**" tab of the results section of the screen.

```
HTTP/1.1 200 OK
Date: Tue, 19 Apr 2011 00:36:58 GMT
Content-Type: application/json; charset=utf-8
X-Powered-By: Servlet/2.5 JSP/2.1
Content-Encoding: gzip
Content-Length: 3003
```

The JSON response data for the above request is provided below for reference:

```
{
```

```
    "STATUS": "SUCCESS",
    "NUM_RECORDS": "67",
    "MESSAGE": "67 records found",
    "AVAILABILITY_UPDATED_TIMESTAMP": "2011-04-18T17:36:58.632-07:00",
    "AVAILABILITY_REQUEST_TIMESTAMP": "2011-04-18T17:36:58.531-07:00",
    "AVL":     [
            {
        "TYPE": "OFF",
        "OSPID": "934",
        "NAME": "Golden Gateway Garage",
        "DESC": "250 Clay Street",
        "INTER": "Clay between Front & Davis",
        "TEL": "(415) 433-4722",
        "OPHRS": {"OPS":          [
                    {
            "FROM": "Monday",
            "TO": "Friday",
            "BEG": "4:00 AM",
            "END": "10:00 PM"
          },
                    {
            "FROM": "Saturday",
            "BEG": "7:00 AM",
            "END": "10:00 PM"
          },
                    {
            "FROM": "Sunday",
            "BEG": "9:00 AM",
            "END": "10:00 PM"
          }
        ]},
        "OCC": "543",
        "OPER": "1100",
        "PTS": "1",
        "LOC": "-122.3986032,37.79544154"
      },
            {
        "TYPE": "ON",
        "BFID": "651001",
        "NAME": "Sansome St (1-99)",
        "OCC": "0",
        "OPER": "0",
        "PTS": "2",
        "LOC": "-122.4007030779,37.790581372,-122.4007936029,37.7910219272"
      },
############# 65 more Availability Records omitted#########
    ]
}
```

However, if incorrect parameters or values are passed the service will return an ERROR status as shown below where longitude is omitted even though the latitude is specified.
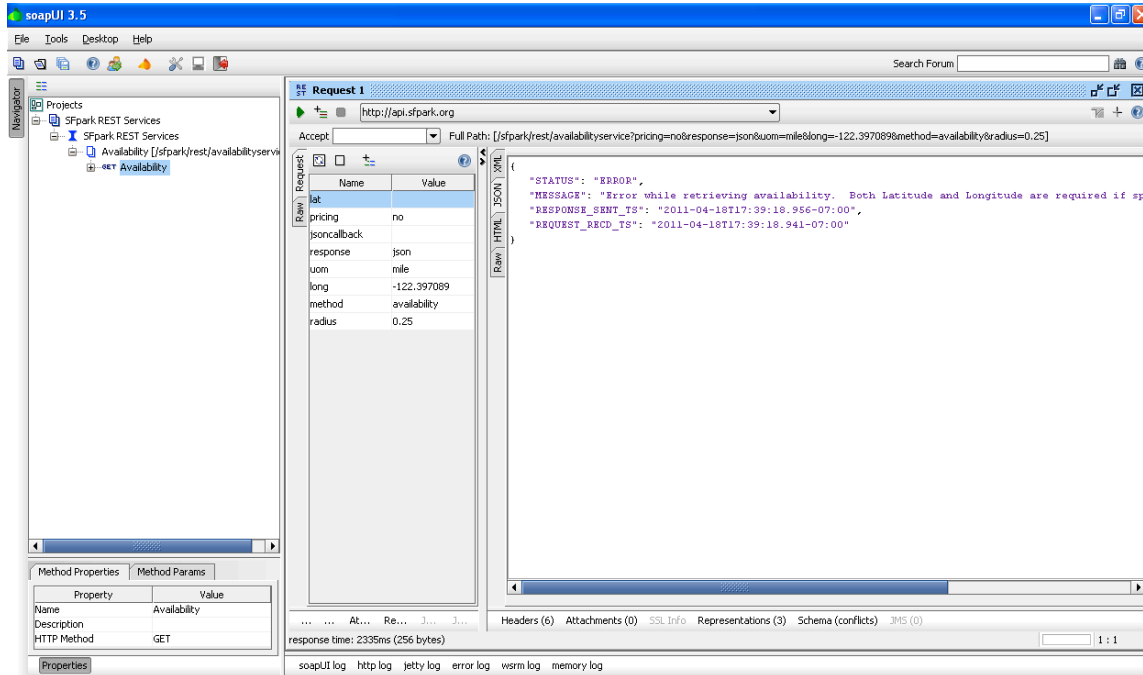
The HTTP status code returned is still 200 since the request processed successfully, since it is data that is not correctly passed. Status code of 500 or 503 will be sent if there is some system error or connectivity issue and the request could not be processed.

The client code should focus mainly on the STATUS element returned in the response as a means of determining if there is a successful return and whether the query passed was correctly processed. The NUM_RECORDS can be used as an easy way to determine number of records returned. This is returned only if the STATUS is SUCCESS. The MESSAGE field may also include number of records returned, however, the NUM_RECORDS field is used primarily for this purpose so it is better suited to use for verifying if any valid records are also returned when STATUS is SUCCESS.

Below is a screen where uom parameter is not correct. Server expects value of mile but miles was sent. It returned in an ERROR status as seen below.
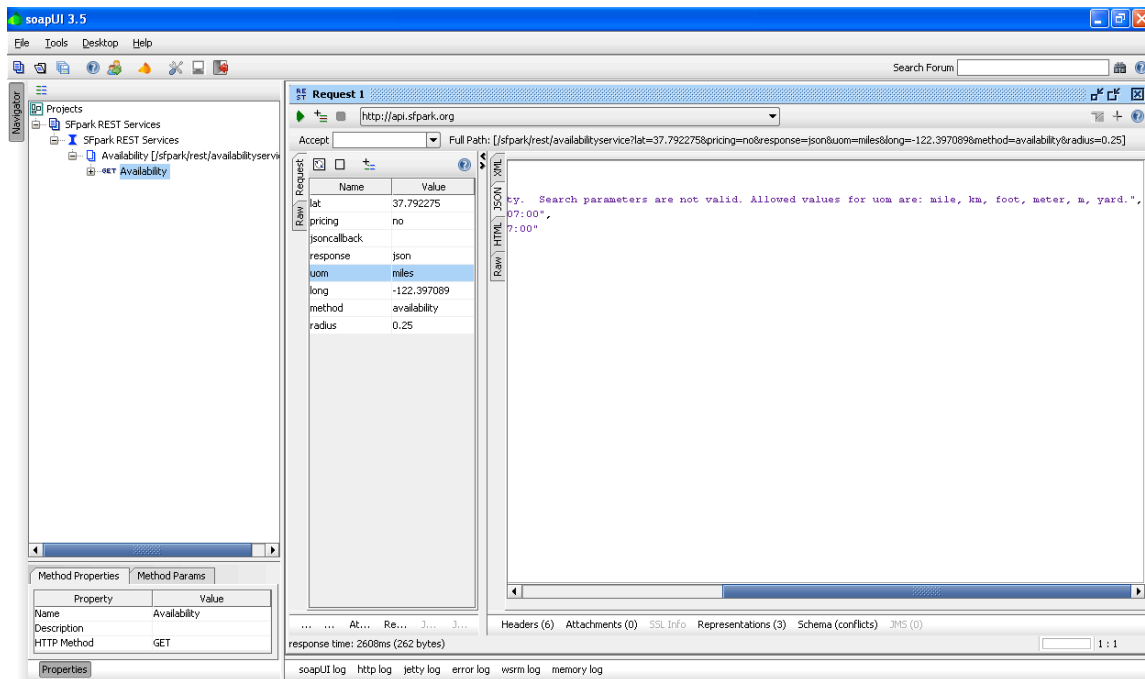
Note that you can clear out all parameter values so that you set only certain ones by clicking the "Rectangle" icon as shown in the screen below. Only the parameters with values typed in will then be passed to the server. You can test with combination of various parameters as appropriate to verify the response.

The service also provides an option to return the data in XML format. This is done by specifying xml (default value) for the response search parameter as shown below. You can then click the XML tab on the search result panel to see the XML formatted response.

If you receive a SUCCESS response similar to below it has successfully returned the availability information for the requested search parameters.

Note: Be sure to review the response in the appropriate result tab, i.e., XML data vs. JSON format as these may look odd when viewed in the other format panel.
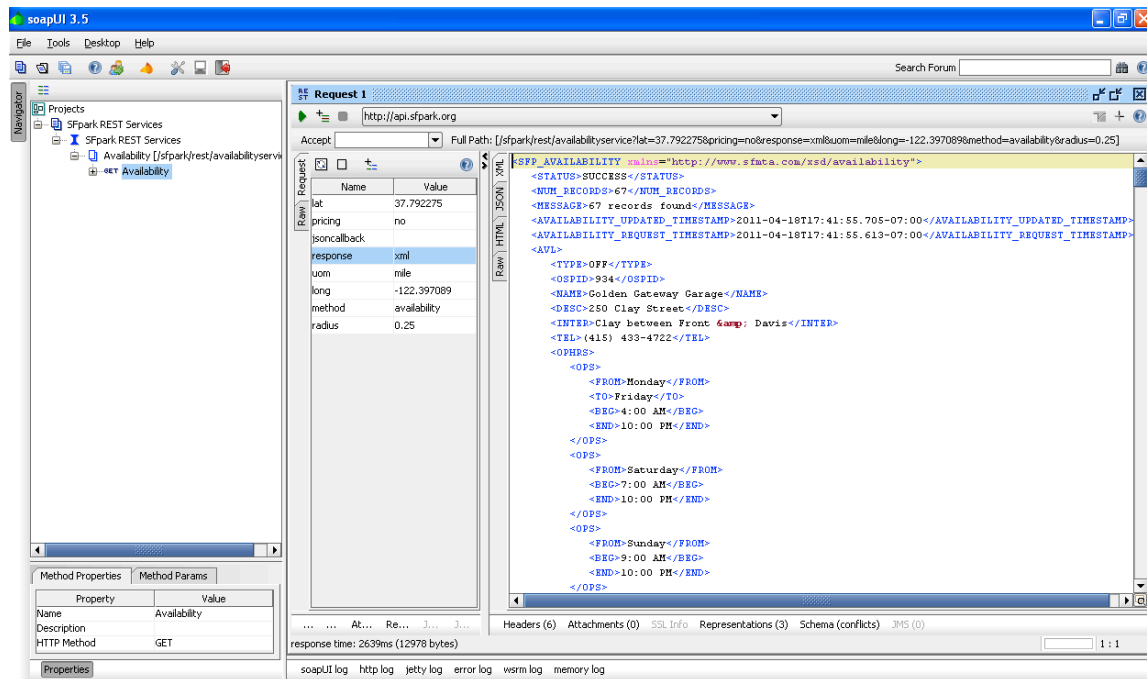
The data returned has been formatted as XML as requested in the search parameter for response format. SUCCESS indicates the search was successfully processed and NUM_RECORDS field indicates the actual number of results found matching the request criteria.

```
<SFP_AVAILABILITY xmlns="http://www.sfmta.com/xsd/availability">
    <STATUS>SUCCESS</STATUS>
    <NUM_RECORDS>67</NUM_RECORDS>
    <MESSAGE>67 records found</MESSAGE>
    <AVAILABILITY_UPDATED_TIMESTAMP>2011-04-18T17:41:55.705-
07:00</AVAILABILITY_UPDATED_TIMESTAMP>
    <AVAILABILITY_REQUEST_TIMESTAMP>2011-04-18T17:41:55.613-
07:00</AVAILABILITY_REQUEST_TIMESTAMP>
    <AVL>
        <TYPE>OFF</TYPE>
        <OSPID>934</OSPID>
        <NAME>Golden Gateway Garage</NAME>
        <DESC>250 Clay Street</DESC>
        <INTER>Clay between Front &amp; Davis</INTER>
        <TEL>(415) 433-4722</TEL>
        <OPHRS>
            <OPS>
                <FROM>Monday</FROM>
                <TO>Friday</TO>
                <BEG>4:00 AM</BEG>
                <END>10:00 PM</END>
            </OPS>
            <OPS>
```

```
            <FROM>Saturday</FROM>
            <BEG>7:00 AM</BEG>
            <END>10:00 PM</END>
        </OPS>
        <OPS>
            <FROM>Sunday</FROM>
            <BEG>9:00 AM</BEG>
            <END>10:00 PM</END>
        </OPS>
    </OPHRS>
    <OCC>518</OCC>
    <OPER>1100</OPER>
    <PTS>1</PTS>
    <LOC>-122.3986032,37.79544154</LOC>
</AVL>
<AVL>
    <TYPE>ON</TYPE>
    <BFID>651001</BFID>
    <NAME>Sansome St (1-99)</NAME>
    <OCC>0</OCC>
    <OPER>0</OPER>
    <PTS>2</PTS>
    <LOC>-122.4007030779,37.790581372,-122.4007936029,37.7910219272</LOC>
</AVL>
 ############ 65 more Availability Records omitted#########
</SFP_AVAILABILITY>
```

Also the HTTP status code of 200 is returned indicated the request processed successfully. This is seen by clicking the "**Raw**" tab of the results section of the screen.

```
HTTP/1.1 200 OK
Date: Tue, 19 Apr 2011 00:43:08 GMT
Content-Type: application/json; charset=utf-8
X-Powered-By: Servlet/2.5 JSP/2.1
Content-Encoding: gzip
Content-Length: 3846
```

Following is an example of JSON response that includes pricing information (pricing =yes)

```
{
    "STATUS": "SUCCESS",
    "NUM_RECORDS": "67",
    "MESSAGE": "67 records found",
    "AVAILABILITY_UPDATED_TIMESTAMP": "2011-04-18T17:43:08.426-07:00",
    "AVAILABILITY_REQUEST_TIMESTAMP": "2011-04-18T17:43:08.142-07:00",
    "AVL":      [
            {
        "TYPE": "OFF",
        "OSPID": "934",
        "NAME": "Golden Gateway Garage",
        "DESC": "250 Clay Street",
```

```
"INTER": "Clay between Front & Davis",
"TEL": "(415) 433-4722",
"OPHRS": {"OPS":          [
                {
        "FROM": "Monday",
        "TO": "Friday",
        "BEG": "4:00 AM",
        "END": "10:00 PM"
    },
                {
        "FROM": "Saturday",
        "BEG": "7:00 AM",
        "END": "10:00 PM"
    },
                {
        "FROM": "Sunday",
        "BEG": "9:00 AM",
        "END": "10:00 PM"
    }
]},
"RATES": {"RS":          [
                {
        "DESC": "Incremental",
        "RATE": "3.5",
        "RQ": "Per 1/2-hr"
    },
                {
        "DESC": "24-Hour Max/Lost Tkt",
        "RATE": "36",
        "RQ": "Flat rate"
    },
                {
        "DESC": "Early Bird",
        "RATE": "20",
        "RQ": "Flat rate",
        "RR": "Mon-Fri: In by 10AM/Out by 7PM"
    },
                {
        "DESC": "Motorcycle",
        "RATE": "7",
        "RQ": "Flat rate"
    },
                {
        "DESC": "Park & Ride Validation",
        "RATE": "3",
        "RQ": "Per day",
        "RR": "Weekend only until 10PM"
    },
                {
        "DESC": "Evening",
        "RATE": "7",
        "RQ": "Flat rate",
        "RR": "Mon-Fri: In after 5PM Out by 8AM"
```

```
            },
                        {
            "DESC": "Weekend",
            "RATE": "7",
            "RQ": "Per day",
            "RR": "Sat: In after 7AM/Out by 4AM;Sun: In after 4AM/Out by
4AM"
            }
        ]},
        "OCC": "514",
        "OPER": "1100",
        "PTS": "1",
        "LOC": "-122.3986032,37.79544154"
    },
            {
        "TYPE": "ON",
        "BFID": "651001",
        "NAME": "Sansome St (1-99)",
        "RATES": {"RS":          [
                        {
            "BEG": "12:00 AM",
            "END": "7:00 AM",
            "RATE": "0",
            "RQ": "No charge"
        },
                        {
            "BEG": "7:00 AM",
            "END": "6:00 PM",
            "RATE": "0",
            "RQ": "Restricted"
        },
                        {
            "BEG": "6:00 PM",
            "END": "12:00 AM",
            "RATE": "0",
            "RQ": "No charge"
        }
        ]},
        "OCC": "0",
        "OPER": "0",
        "PTS": "2",
        "LOC": "-122.4007030779,37.790581372,-122.4007936029,37.7910219272"
    },
############# 65 more Availability Records omitted#########
    ]
}
```