

System Overview

Facility Source (FS) provides facility maintenance optimization to its Clients nationwide. We use **fmPilot 1.0 & fmPilot 2.0** as a proprietary software to facilitate every aspect of our service.

Both the systems offer our Clients a one stop solution for **work order Management** and **Asset Tracking**. These systems are used by our Associates, Clients and Service Provider (SP's).

Each user has permission based access, so they have access to functions and visibility within the fmPilot systems based on their requirement.

fmPilot has two parts: 1.0 and 2.0, and usage of these depends on the SP chosen to complete the work.



fmPilot: This system is used by our Clients to create and manage work order (WO).

fsElite: This app is used to track the SP and their activities on the assigned WO.

Service Provider API

The SP API is a web service exposing a set of interfaces and targeting common needs of the SP community, building a bridge between FS fmP2 system and the SP IT proprietary systems. The SP API functionality is focused on WO Management and Invoice/Quote Management.

The three operations most requested by our SP's are WO pulling, WO updates (including WO status), and the ability to add comments or notes to WO. In each of these cases, the primary concern is to make these operations available to SP's primary systems. These are not expected to be made available to individual distributed applications where user/role security is necessary. Each action will behave as if it is being performed by a SP administrator level core system.

Work Order Pulling

The first required operation is pulling of existing WO to retrieve assigned work orders and update the existing work orders. This is an on-demand service that will respond to requests for data rather than an active push of new data, which would require SP's to construct their own services for reception. There are finite limits on the numbers of work orders that may be retrieved in a single pull to prevent any resource "hogging" by large queries.

Work Order Updates

SP frequently have a need to modify the status of a WO, and often the status has been modified within a SP's core systems and that data simply needs to be transmitted. This may be a fulfillment of a WO, acceptance of a WO, or suspense of a WO that is awaiting materials.

Each of these statuses has its own business logic within the application itself, so this will need to be mirrored within the service to ensure that all notifications, validations and processes are followed correctly. Additionally, all status updates undergo a general validation rule prior to transition-specific validations.

Work Order Comments and Notes

As WO evolve, technicians need to apply notes either for client usage, notes to continuing technicians, and other pertinent administrative information. A service is implemented that receives and annotates these comments relative to a particular WO. Pulling for comments is also implemented.

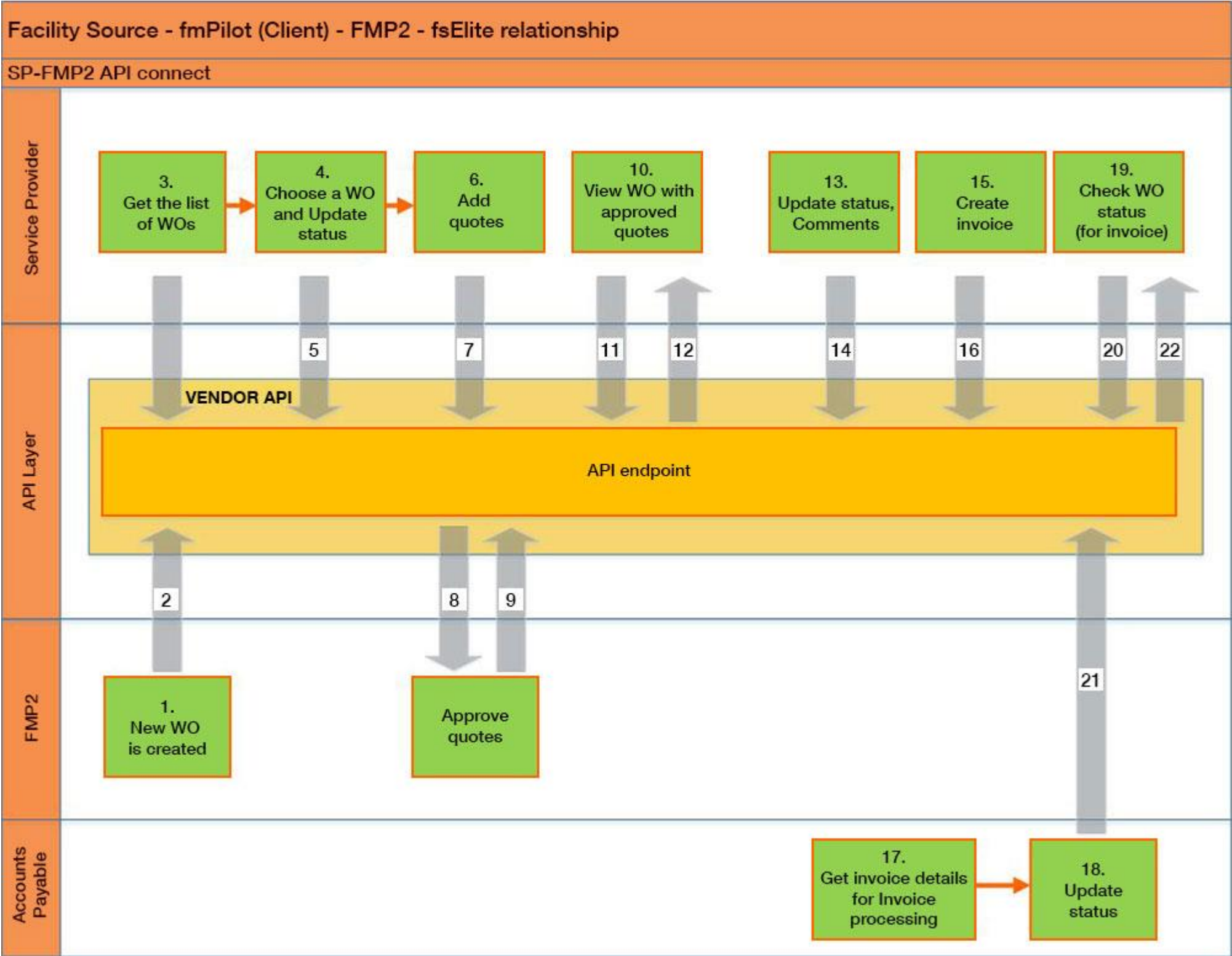
Quote

Quote management functionality is exposed to SP through API(s) where SP can get the quotes associated with work order, create/update the quote, insert attachments to the specified quote and so on.

Invoice

Invoice management functionality is exposed to SP through API(s) where SP can get the invoice(s) associated with work order, create/update the invoice, insert attachments to the specified invoice and so on.

System Workflow



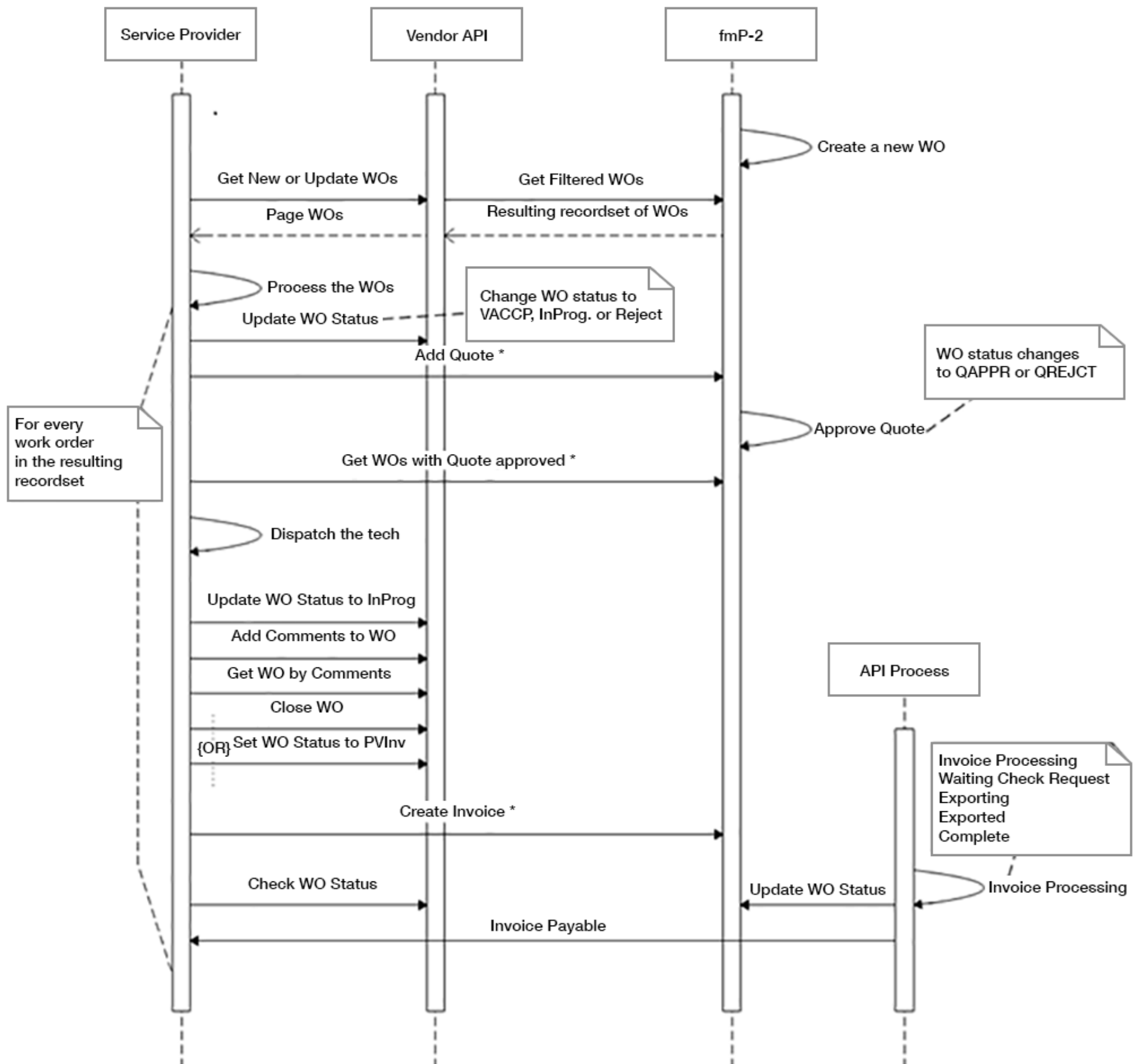
Different steps involved in the above diagrams are-

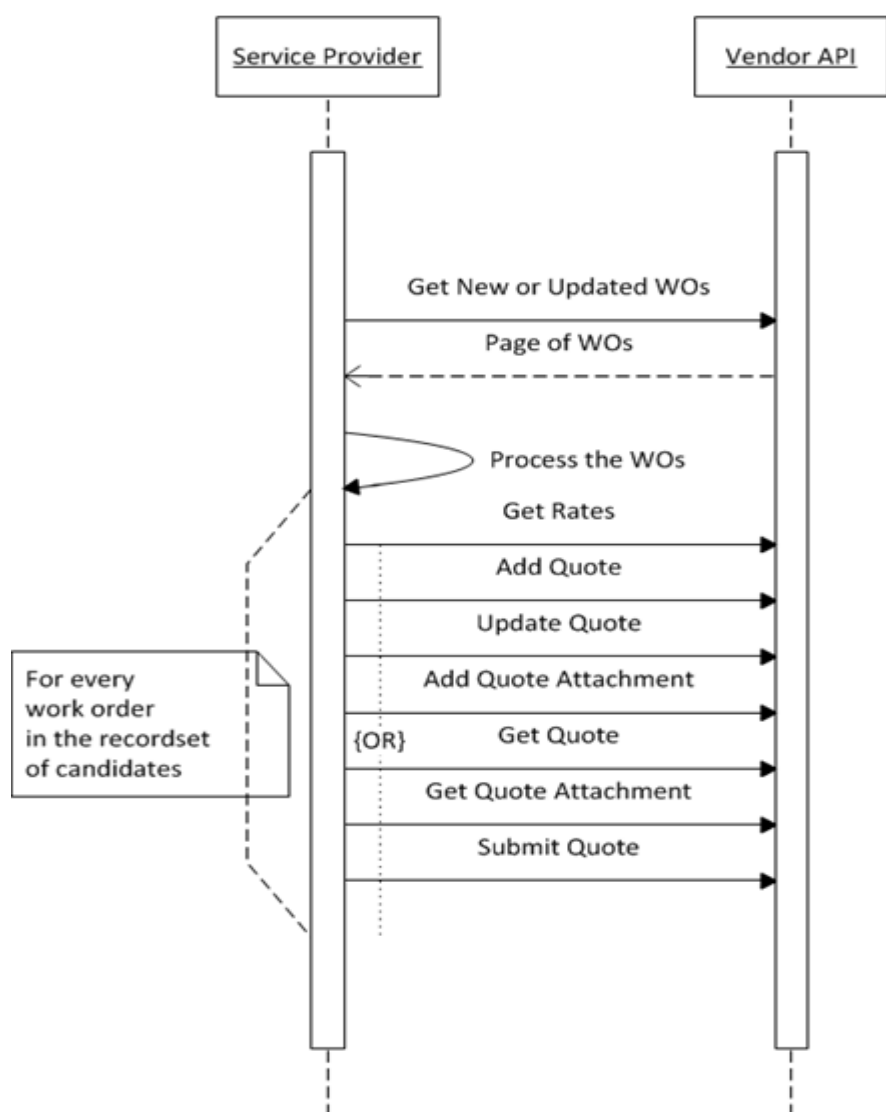
1. A new work order (WO) is created in FMP2 (either directly or through fmPilot).
2. The FMP2 hits the API endpoint and passes WO details to be updated in the database by the SP.
3. Service Provider (SP) chooses to view the WO list.
4. SP API sends the list of WOs to SP (fsElite).
5. SP chooses a WO and updates the status. The WO status is sent to the SP API endpoint that updates the new status in the database.
6. SP adds quotes for the WO.
7. The quotes added are sent to the API endpoint, which updates the database.
8. FMP2 user requests to see the quotes for providing subsequent approval.
9. FMP2 user approves the quotes and the approval flag is conveyed through API into the database.
10. SP (fsElite) requests to see the quotes decision.
11. fsElite makes a call to the API to see the quotes decision.
12. API provides the approved quotes information to SP (fsElite).
13. SP works on the WO (offline) and keeps updating comments, attachments etc.
For e.g. A SP completes a WO due to which status of WO changes from In Progress to Pending SP Invoice.
Once this change in status happens in fsElite, this needs to be synced up in client system using the set of APIs aligned with this task.
14. Each of the updates to the WO are sent to the API endpoint and API updates the FMP2 database.
15. Finally, when the work is completed, SP creates the invoice.
16. Invoice draft is provided by the fsElite to the API endpoint to be updated in the database.
17. Account and Payable/ team query the database for the WO invoice details.
18. Views the invoice info and updates status (approved/rejected).

19. SP attempts to see the status of the provided invoice on his WO.
20. fsElite queries the WO status for an invoice using the API call.
21. API fetches the WO invoice details from the database.
22. API provides the fetched WO invoice decision to the SP (fsElite).

System Architecture

The system diagram below displays a high-level interaction between: the Service Provider (SP), Facility Source's (FS) IT systems and their respective Clients. The FS IT system provides a platform for work order (WO) management, which is integrated with SP fsElite system and the FS Client system. The interactions usually happen through different API endpoints that manage the WO life cycle. This maintains a sync between different systems.





Implementation Consideration

Before you build an integration application or other client application, consider the data management, use limits, and communication issues explained in this section.

Web API description

The system implements HTTP services by employing ASP.NET API Web API that can be consumed by a broad range of clients including browsers, mobiles, phones and tablets. It supports variety of **Get, Put and Post** request-response message system and provides the result in **JSON/XML** format.

Recommended testing tools

One of the simplest tools for testing is Postman. It is currently one of the most popular tools used in API testing. We will be referencing this tool mostly, but any API testing tool can be used with similar configuration and results.

Before going live, thoroughly test and debug your applications in the Sandbox Mode. You can use your favorite REST client to explore our API, but we recommend the [Postman](#), or [Fiddler](#).

Content negotiation

With Web API content negotiation, the system returns data based on the client requests. If the client is requesting the data to be returned as JSON or XML, the Web API framework deals with the request type and returns the data appropriately based on the media type. By default Web API provides JSON based response.

API Overview

The SP API is a RESTful and stateless API. It has predictable, resource-oriented URLs and accepts POST, GET, PUT, and DELETE requests. Our API uses standard HTTP response codes and returns JSON-encapsulated data.

API Version

The current stable version of the SP API is **v1**

Environments

There are two [environments](#) for working with our API: Sandbox for testing and Production for live integrations.

App Registration

To use our API, you should have either a subscriber account with the Super Admin role or provider account with admin rights. You also need to [register](#) your application.

Authentication and Authorization

We use token authentication, yet secure and effective [authentication and authorization](#)

You must include the Authorization header in every API request. Unauthorized calls will fail.

Requests and Responses

Making requests are easy: our API accepts the most common HTTP verbs and has well-structured endpoint routes.

After a successful request, you will receive a JSON-formatted response as well as a standard [HTTP status code](#).

Call Limits

There are no limits to how many calls you can make in a minute.

Sandbox Mode

We strongly recommend you to start in Sandbox Mode and move to the Production Environment only when your application is fully tested and ready to go live.

App Registration

Register your application to obtain calling client, acting domain and authentication token that are required for authentication and authorization.

To test an application in the Sandbox or Production Environment:

Currently staging environment is a sandbox environment.

To test API(s) use `https://staging-api.fmpilot2.com/Vendor/api/{{apiname}}`

for e.g. to test clients domains execute - `"https://staging-api.fmpilot2.com/Vendor/api/Clients/Domains/"`

Security

The system implements HTTP services by employing ASP.NET WebAPI that can be consumed by a broad range of clients including browsers, mobiles, phones and tablets.

It supports variety of

GET, PUT and POST

 request-response message system and provides the result in JSON or XML format.

With WebAPI content negotiation, the system returns data based on the client requests. If the client is requesting the data to be returned as JSON or XML, the WebAPI framework deals with the request type and returns the data appropriately based on the media type. By default, WebAPI provides JSON based response.



WebAPI is a trending technology. As we are exposing our WebAPI to the outside world, we should maintain security in WebAPI. It means a valid user can only access WebAPI, or else it will throw an unauthorization error.

Authentication


One of our primary concerns will be security and integrity of data and/or requests, For more details proceed to [Authentication and Authorization](#).

Authentication

One of our primary concerns will be security and integrity of data and/or requests, so we will be establishing a special API authorization that will be used to perform an identity challenge along with domain to verify that the SPs are who they say they are.

Basic authentication is a simple authentication scheme built into the HTTP protocol. If a request requires authentication, the server returns 401 (Unauthorized). The response includes a WWW-Authenticate header, indicating the server supports Basic authentication.



 WebAPI is a trending technology. As we are exposing our WebAPI to the outside world, we should maintain security in WebAPI. It means a valid user can only access WebAPI, or else it will throw an unauthorization error.

Error Handling

The API calls return error data that your client application can use to identify and resolve runtime errors. If an error occurs during the invocation of most API calls, then the API provides the following types of error handling:

For errors resulting from badly formed messages, failed authentication, or similar problems, the API returns a fault message with an associated Exception Code.

For most calls, if the error occurs because of a problem specific to the query, the API returns an Error.

Exception Handling

What happens if a WebAPI controller throws an uncaught exception? By default, most exceptions are translated into an HTTP response with status code 500, Internal Server Error. The `HttpResponseException` type is a special case. This exception returns any HTTP status code that you specify in the exception constructor. For example, the following method returns 404, Not Found, if the id parameter is not valid.

Sample:

```
public Product GetProduct(int id) {
    Product item = repository.Get(id);
    if (item == null)
    {
        throw new HttpResponseException(HttpStatusCode.NotFound);
    }
    return item;
}
```

In case of failure, the controller methods throw an `HttpResponseException` exception with the appropriate error message, which allows for a given `HttpResponseMessage` to be returned to the client.

API Fault Element

Fault	Description
NO_WO_NUMBER_OR_ID	The requested action requires a WO number or id.
INVALID_QUOTE_ID	The requested action requires a valid quote id.
INVALID_INVOICE_ID	The requested action requires a valid invoice id.
EXISTING_INVOICE	The WO already has an active invoice.

Fault	Description
INVALID_QUOTE_ORDER_ID	The requested action requires a quote order number 1 or 2.
WO_NOT_FOUND	The WO requested could not be found with the specified parameters.
NO_COMMENT_SUPPLIED	No comment was included to the request. A comment is required for this operation.
NO_WO_SUPPLIED	No WO was included to the request. A WO is required for this operation.
INVALID_DNE_UPDATE	DNE value cannot be changed.
INVALID_ACTUAL_DATE_UPDATE	An actual date value cannot be changed.
INVALID_MODIFIED_DATE_RANGE	Filtering by Modified DateTime requires Equals comparison or setting a Range of Dates.
INVALID_STATUS_CODE	StatusCode value is not valid.
INVOICE_NOT_FOUND	The invoice could not be found.
TRIPCHARGE_RATE_NOT_FOUND	The trip charge rate could not be found.
INVALID_ATTACHMENT_EXTENSION_TYPE	The attachment extension type {0} is not valid. Valid extension type is: {1}
QUOTE_NOT_FOUND	No quotes could be found for the supplied WO.
NO_QUOTE_SUPPLIED	No quote was included to the request. A quote is required for this operation.

Fault	Description
NO_INVOICE_SUPPLIED	No invoice was included to the request. An invoice is required for this operation.
INVALID_PVINV_STATUS	The action requires PVINV WO status.
INVALID_PQTE_STATUS	The action requires PQTE WO status.
NO_FILTER_CRITERIA	The requested action requires a filter criteria.
NO_VENDOR_CODE	Please provide the calling client as part of the Header information.

Response Codes

Success Message(s)

Code	Text	Description
200	OK	The request was successful.
202	Accepted	The request has been accepted, but not yet processed.
203	Non-Authoritative Information	The request was successful, but the response data may be from a third party.

Redirection

Code	Text	Description
300	Multiple Choices	The request has more than one representation options, for example, different file formats. Select a preferred representation and redirect your request to the representation location.
301	Moved Permanently	The resource you requested has been permanently moved to a new location. Direct this and all future requests to the Uniform Resource Identifier (URI) specified in the Location response header.
302	Found	The resource you requested has been temporarily moved to a new location. Direct this request to the temporary URI specified in the Location response header, but continue using the original URI for future requests.
303	See Other	You can find the response to your request under another URI. Send a GET request to the URI specified in the Location response header.
304	Not Modified	The resource has not been modified since last requested, so there is no new data to return.
305	Use Proxy	Access the requested resource through the proxy provided in the Location response header.
307	Temporary Redirect	The resource you requested has been temporarily moved to a new location. Direct this request to the temporary URI specified in the Location response header, but continue using the original URI for future requests. In contrast to 302, you cannot change the request method when reissuing the original request.

Client Error(s)

Code	Text	Description
400	Bad request	The request was not accepted due to bad syntax, missing parameters, insufficient data, etc.
401	Unauthorized	Missing or incorrect authentication credentials.

Code	Text	Description
403	Forbidden	You are not authorized to request this resource, or the resource is unavailable for some reason.
404	Not Found	The request URI is incorrect, or the resource does not exist.
405	Method Not Allowed	The method specified in the Request-Line is not allowed for the resource identified by the Request-URI. The response MUST include an Allow header containing a list of valid methods for the requested resource.
406	Not Acceptable	This code is similar to 401 (Unauthorized), but indicates that the client must first authenticate itself with the proxy. The proxy MUST return a Proxy-Authenticate header field (section 14.33) containing a challenge applicable to the proxy for the requested resource. The client MAY repeat the request with a suitable Proxy-Authorization header field (section 14.34). HTTP access authentication is explained in "HTTP Authentication: Basic and Digest Access Authentication".
407	Proxy Authentication Required	The format specified in the Accept header is not supported. Usually, the possible format is JSON. See Output Formats for details.
408	Request Timeout	The server has timed out while waiting for the request. You can repeat the request without modifications at any later time.
409	Conflict	The server was not able to process the request because of a conflict. Study the response body to recognize the source of the problem. Resolve the conflict and reissue the request.
410	Gone	The resource you requested is no longer available. Do not request this resource again in the future.
411	Length Required	You have not stated the length of the content, which is required by the server. Add a valid Content-Length header and repeat the request.
412	Precondition Failed	The server could not meet one or more of the preconditions stated in the request headers.
413	Request Entity Too Large	The request is larger than the server is willing or able to process.

Code	Text	Description
414	Request-URI Too Long	The URI provided is longer than the server is willing or able to interpret. You have probably used too many query-strings in a GET request. In this case, try to convert it into a POST request.
415	Unsupported Media Type	The request contains a media type that the server or resource does not support.
416	Requested Range Not Satisfiable	The server cannot provide the portion of the data that you specified in the Range request header. It is possible that the range is outside the size of the target data.
417	Expectation Failed	The server could not meet the expectation stated in the Expect request header.
421	Misdirected Request	The request was directed at a server that is not able to produce a response. This can be sent by a server that is not configured to produce responses for the combination of scheme and authority that are included in the request URI.
423	Locked	The resource that is being accessed is locked.
424	Failed Dependency	The request failed due to failure of a previous request.
425	Too Early	Indicates that the server is unwilling to risk processing a request that might be replayed.
426	Upgrade Required	The server refuses to perform the request using the current protocol but might be willing to do so after the client upgrades to a different protocol. The server sends an Upgrade header in a 426 response to indicate the required protocol(s).
428	Precondition Required	Switch to a protocol stated in the Upgrade header and repeat the request.
429	Too Many Requests	The user has sent too many requests in a given amount of time ("rate limiting").

Code	Text	Description
431	Request Header Fields Too Large	The server is unwilling to process the request because its header fields are too large. The request MAY be resubmitted after reducing the size of the request header fields.
451	Unavailable For Legal Reasons	The user requests an illegal resource, such as a web page censored by a government.

Server Error(s)

Code	Text	Description
500	Internal Server Error	Something went wrong, and the server was unable to complete your request. Should this problem persists.
503	Service Unavailable	The server is currently unavailable, or you have reached the throttling limit. Throttling-To keeps our API healthy and protect it from overuse, we set limits on the number of calls you can make per time interval. When developing your app, make sure to respect the API call limits.
504	Gateway Timeout	The server, while acting as a gateway or proxy, has waited too long for a response from the upstream server.
505	HTTP Version Not Supported	The server does not support the HTTP version used in the request.
506	Variant Also Negotiates	The server has an internal configuration error: transparent content negotiation for the request results in a circular reference.
507	Insufficient Storage	The server has an internal configuration error: the chosen variant resource is configured to engage in transparent content negotiation itself, and is therefore not a proper end point in the negotiation process.
508	Loop Detected	The server detected an infinite loop while processing the request.

Code	Text	Description
510	Not Extended	Further extensions to the request are required for the server to fulfill it.
511	Network Authentication Required	The 511 status code indicates that the client needs to authenticate to gain network access.

Object Basics

Generally speaking, API objects represent database tables that contain your organization's information. For example, the central object in the Facility Source (FS) data model represents accounts—companies and organizations involved with your business, such as customers, partners, and competitors. The term “record” describes a particular occurrence of an object. A record is analogous to a row in a database table. Objects can be standard objects, custom objects and external objects.

Applications work with only the objects that you are authorized to access. Programmatic access to objects is determined by the objects defined in your organization, your organization configuration, your user permissions and access settings (which are configured by your organization’s system administrator), your data sharing model, and other factors related specifically to the object.

Primitive Data Types

The API uses the following primitive data types:

Value	Details
Base64	Base 64-encoded binary data. Fields of this type are used for storing binary files in Attachment records, Document records, and Scontrol records.
boolean	Boolean fields have one of these values: true (or 1), or false (or 0).
Byte	A set of bits.
Date	Fields of this type contain date values, such as ActivityDate in the Event object.
DateTime	Fields of this type handle date/time values (timestamps), such as ActivityDateTime in the Event object or the CreatedDate, LastModifiedDate, or SystemModstamp in many objects.

Value	Details
Double	Fields of this type can contain fractional portions (digits to the right of the decimal place), such as ConversionRate in CurrencyType. Scale: Maximum number of digits to the right of the decimal place. Precision: Total number of digits, including those to the left and the right of the decimal place.
int	Fields of this type contain numbers with no fractional portion.
string	Fields that are of data type string contain text and some have length restrictions depending on the data being stored. The MailingStreet is 255 characters.
time	Fields of this type handle time values, such as FridayEndTime in the BusinessHours object. Development tools differ in the way that they handle time data.

Fields types:

Comments Version

Value	Details
CommentFragment	A Collection of data See all collection list

List of CommentFragment

Name	Type
Id	integer
Subject	string
Body	string

Name	Type
WorkOrderNumber	string
OwnerUserId	integer
CreateUser	string
CreateDateTime	date
AllowInternal	boolean
AllowClient	boolean
AllowVendor	boolean
AllowCorporate	boolean
WorkOrderStatusID	integer
CommentTypeID	integer
IsEmailHighImportance	boolean
RequestCodeID	integer
User	string

Name	Type
Role	string
Company	string
FirstName	string
LastName	string

Filtering

Value	Detail
PartialFilterClauseModel	A Collection of data See all collection list

List of PartialFilterClauseModel

Name	Type
Operation	KeywordOperation
NegateOperation	boolean
Keyword	string

List of KeywordOperation

Possible enumeration values:

Name	Value
Equal	0
Contain	1
LessThan	2
LessThanOrEqual	3
GreaterThan	4
GreaterThanOrEqual	5
StartsWith	6

WorkOrdersVersion1

Value	Details
ExternalWorkOrder	A Collection of data See all collection list
ExternalAttachment	A Collection of data See all collection list

List of ExternalWorkOrder

Name	Type
Id	integer
WorkOrderNumber	string

Name	Type
Type	string
WorkType	string
Status	string
StatusCode	string
Description	string
Priority	string
DateReported	date
DateModified	date
TargetDate	date
ScheduledStartDateTime	date
ScheduledCompleteDateTime	date
ActualStartDateTime	date
ActualCompleteDateTime	date

Name	Type
LatestInteractiveVoiceResponseIn	date
LatestInteractiveVoiceResponseOut	date
GLCode	string
RequestingContact	string
AlternateContact	string
RequestTypeId	integer
RequestType	string
DepartmentId	integer
Department	string
TradeId	integer
RequestCodeId	integer
RequestCode	string
ServiceLocationName	string

Name	Type
DNE	decimal number
WorkTypeGLCode	string
RequestGLCode	string
DispatchMatrixGLCode	string
AlternateWorkOrderNumber	string
IsETAMissed	boolean
MissedETAMessage	string
IsEmergency	boolean
IsVendorOnSite	boolean
HasAttachments	boolean
TargetCompleteDate	date
IsSnow	boolean
ProjectNumber	string

Name	Type
ProjectOther	string
CurrencyCode	string

List of ExternalAttachment

Name	Type
WorkOrderNumber	string
Id	integer
InvoiceId	integer
QuoteId	integer
Name	string
Path	string
URL	string
Username	string
CreatedDate	date

Name	Type
FormattedCreateDate	string
SizeString	string
Latitude	string
Longitude	string

Environments

SP API provides two environments: Sandbox Mode and Production.

Sandbox Mode

The SP is invited to join the sandbox, which currently is our Staging environment. As a part of SP invitation to Service Provider API in Staging, we issue an authorization token to this SP. Then, we send a letter to the SP API manager so he would send it to the SP.

Execute APIs:

Follow [document](#) to execute APIs.

Production

We strongly recommend you to start in Sandbox Mode and move to the Production Environment only when your application is fully tested and ready to go live. Production is an environment that contains real data and is designed for live integrations. Use this environment only for mature applications that are 100% ready to go live.

Getting Started

Facility Source (FS) API has a host of different features, but the cornerstone of our platform is a

work order

.

About a Work Order

Think of a work order (WO) as a task issue. A WO details what happened and what should be done; it has a due date, assignee, and much more.

When subscribers need services like AC repair, plumbing etc, they create WO's that are assigned to certain providers. These providers solve the problem and “close” the WO.

Required Parameters

To create a WO, you need to pass some required information about the problem that occurred. In real life, you collect all these values using our API, but for a quick start we are providing you with the ready data.

Parameter	Description	Value
LocationNumber	The location number or FMP2 location ID	string
RequestingContact	Person who is requesting the work	string
AlternateContact		string
Department		string
RequestType	Work Request Type or FMP2 Request Type ID	string
RequestCode	Work Request Code or FMP2 Request Code ID	string
ServiceLocation	Place where the issue exists or FMP2 Service Location ID	string
Description	Description of the needed work	string
AlternateWorkOrderNumber	Optional Value: Defines the external WO reference to link this entity.	string

Clients

Main entry point for managing comments

API	Description
GET api/Clients/Domains	No documentation available.
POST api/ClientsVersion1?key={key}	No documentation available.

Comments

Main entry point for managing comments

API	Description
GET api/Comments	Get the list of comments based on a given filtering

API	Description
GET api/Comments/Count	Gets the count of comments associated.
POST api/CommentsVersion1?key={key}	No documentation available.

Filtering

API	Description
POST api/Filtering	Post method used for getting the serialize filtering query string
POST api/Filtering/Deserialize?serializedClauses={serializedClauses}	Post method used for getting the deserialize value from query string
POST api/Filtering?key={key}	No documentation available.

WorkOrders

Main entry point for managing and retrieving workorder information.

API	Description
GET api/WorkOrders	Gets a list of workorders considering a given criteria
GET api/WorkOrders/{id}	Gets a work order by workorder number
GET api/WorkOrders/Count	Gets the total workorder count.
PUT api/WorkOrders/{id}/Reject	No documentation available.
GET api/WorkOrders/WorkOrderNumbers	Get work orders list by client
GET api/WorkOrders/{id}/Attachments	Gets the attachments associated with the specified workorder and invoice
POST api/WorkOrders/{id}	Modifies a workorder based on the provided attributes

API	Description
POST api/WorkOrders/{id}/Attachments	Inserts the associated attachments in the POST Request to the specified workorder
GET api/WorkOrders/{id}/Equipments	Gets the equipments valid for the specified workorder and invoice
GET api/WorkOrders/{id}/IVRLog	No documentation available.
GET api/WorkOrders/{id}/WorkOrderStatus	Gets the Workorder status associated to the specified workorder
GET api/WorkOrders/{id}/Comments	Gets the list of comments associated with a workorder
PUT api/WorkOrders/{id}/Comments	Adds a new comment associated to a given workorder
GET api/WorkOrders/{id}/Invoice	Gets the invoice associated to the specified workorder
POST api/WorkOrders/{id}/Invoice/{invoiceId}/Attachments	Inserts the associated attachments in the POST Request to the specified workorder and invoice
POST api/WorkOrders/{id}/Invoice	Adds an Invoice associated to a given WorkOrder
GET api/WorkOrders/{id}/Invoice/{invoiceId}/Attachments	Gets the attachments associated with the specified workorder and invoice
POST api/WorkOrders/Invoice/Bulk	No documentation available.
PUT api/WorkOrders/{id}/Invoice/{invoiceId}	No documentation available.
PUT api/WorkOrders/{id}/Invoice/Submit	No documentation available.
GET api/WorkOrders/{id}/Quotes	TFS Task 11630. Gets the quotes associated to the specified workorder
GET api/WorkOrders/{id}/Quotes/{quoteOrder}	Gets the quotes associated to the specified workorder
POST api/WorkOrders/{id}/Quotes	No documentation available.
PUT api/WorkOrders/{id}/Quotes/{quoteId}	No documentation available.
GET api/WorkOrders/{id}/Quotes/{quoteId}/Attachments	Gets the attachments associated with the specified workorder and quote
POST api/WorkOrders/{id}/Quotes/{quoteId}/Attachments	Inserts the associated attachments in the POST Request to the specified workorder and quote
PUT api/WorkOrders/{id}/Quotes/Submit	No documentation available.

API	Description
GET api/WorkOrders/{id}/Rates	No documentation available.
GET api/WorkOrders/{id}/Taxes	Gets the possible tax information associated to the provided WorkOrder
GET api/WorkOrders/{id}/CauseRemedy	Gets the cause/remedy options for the provided workorder and relaed RequestCode
POST api/WorkOrdersVersion1?key={key}	No documentation available.

BaseVendorAPI

API	Description
POST api/BaseVendorAPI?key={key}	No documentation available.

.Net Sample

Add authentication information to request headers

GetHttpClient(), GetAsync(),GetAsyncCall() are defined in WebAPIProxy class
Sample:

```
private string baseAddress = "http://demo-api.fmpilot2.com/Vendor/api/"

private const string authTokenKey = "authenticationToken";

private const string clientKey = "callingClient";

private const string domainKey = "actingDomain";


private HttpClient GetHttpClient(string authTokenValue, string clientValue, string domainValue, string contentType)
{
    HttpClient client = new HttpClient();

    // Set the Header values

    client.DefaultRequestHeaders.Accept.Clear();

    client.DefaultRequestHeaders.Add(authTokenKey, authTokenValue);

    client.DefaultRequestHeaders.Add(clientKey, clientValue);

    client.DefaultRequestHeaders.Add(domainKey, domainValue);

    client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue(contentType));


    // Set the base address

    client.BaseAddress = new Uri(baseAddress);
```

```
        return client;
    }
}
```

Creating WorkOrder Object

Sample:

```
public class WO
{
    public string WorkOrderNumber { get; set; }

    public string Type { get; set; }

    public string Status { get; set; }

    public string Description { get; set; }

    public string Priority { get; set; }

    public DateTime? DateReported { get; set; }

    public DateTime? DateModified { get; set; }

    public DateTime? TargetDate { get; set; }

    public DateTime? ScheduledStartDateTime { get; set; }

    public DateTime? ScheduledCompleteDateTime { get; set; }

    public DateTime? ActualStartDateTime { get; set; }

    public DateTime? ActualCompleteDateTime { get; set; }

    public string RequestingContact { get; set; }

    public string AlternateContact { get; set; }

    public string RequestType { get; set; }

    public string RequestCode { get; set; }

    public decimal DNE { get; set; }

    public Location Location { get; set; }

    public Equipment Equipment { get; set; }

    public Cause Cause { get; set; }

    public Remedy Remedy { get; set; }
}
```

Creating Result Class

Sample:

```
public class WebApiResponse<T>
{
    public System.Net.HttpStatusCode StatusCode { get; set; }

    public string ErrorMessage { get; set; }

    public T Result { get; set; }
}
```

Executing WorkOrder(GET api/WorkOrders/{id})

Sample:

```
WebAPIProxy webAPIProxy = new WebAPIProxy();

WebApiResult<WO> result = webAPIProxy.GetAsync<WO>(authTokenValue, clientValue, domainValue, "WorkOrders/" + txtWONumber.Text,
"application/json");

public WebApiResult<T> GetAsync<T>(string authTokenValue, string clientValue, string domainValue, string actionName,string contentType)

{

    WebApiResult<T> output = new WebApiResult<T>();

    try

    {

        var result = GetAsyncCall<T>(authTokenValue, clientValue, domainValue, actionName, contentType);

        result.Wait();

        return result.Result;

    }

    catch(Exception e)

    {

        output.StatusCode = System.Net.HttpStatusCode.InternalServerError;

        output.ErrorMessage = NoErrorMessage;

        return output;

    }

}

private async Task<WebApiResult <T>>> GetAsyncCall<T>(string authTokenValue, string clientValue, string domainValue,string actionName, string contentType)

{

    Func <Task<WebApiResult <T>>>> valueFactory = async () =>

    {

        WebApiResult<T> output = new WebApiResult<T>();

        HttpClient client = GetHttpClient(authTokenValue, clientValue, domainValue, contentType);

        HttpResponseMessage response = await client.GetAsync(actionName).ConfigureAwait(false);

        output.StatusCode = response.StatusCode;

        if (response.StatusCode == System.Net.HttpStatusCode.OK)

            output.Result = await response.Content.ReadAsAsync<T>();

        else

        {

            output.ErrorMessage = await response.Content.ReadAsStringAsync();

            if (string.IsNullOrEmpty(output.ErrorMessage))

            {

                output.ErrorMessage = NoErrorMessage;

            }

        }

    }

}
```

```
return output;

};
```

Abbreviations

List of Abbreviations:

Service Provider	SP
Work Order	WO
Facility Source	FS
Open Authorization	OAuth
Uniform Resource Identifier	URI
Application Program Interface	API
Information Technology	IT
Hypertext Transfer Protocol	HTTP
Secure Sockets Layer	SSL
Air Conditioning	AC
Representational State Transfer	REST

JavaScript Object Notation	JSON
Uniform Resource Locator	URLs
eXtensible Markup Language	XML