API: Getting Started

EPICOR.

Disclaimer

This document is for informational purposes only and is subject to change without notice. This document and its contents, including the viewpoints, dates and functional content expressed herein are believed to be accurate as of its date of publication. However, Epicor Software Corporation makes no guarantee, representations or warranties with regard to the enclosed information and specifically disclaims any applicable implied warranties, such as fitness for a particular purpose, merchantability, satisfactory quality or reasonable skill and care. As each user of Epicor software is likely to be unique in their requirements in the use of such software and their business processes, users of this document are always advised to discuss the content of this document with their Epicor account manager. All information contained herein is subject to change without notice and changes to this document since printing and other important information about the software product are made or published in release notes, and you are urged to obtain the current release notes for the software product. We welcome user comments and reserve the right to revise this publication and/or make improvements or changes to the products or programs described in this publication at any time, without notice.

The usage of any Epicor software shall be pursuant to an Epicor end user license agreement and the performance of any consulting services by Epicor personnel shall be pursuant to Epicor's standard services terms and conditions. Usage of the solution(s) described in this document with other Epicor software or third party products may require the purchase of licenses for such other products. Where any software is expressed to be compliant with local laws or requirements in this document, such compliance is not a warranty and is based solely on Epicor's current understanding of such laws and requirements. All laws and requirements are subject to varying interpretations as well as to change and accordingly Epicor cannot guarantee that the software will be compliant and up to date with such changes. All statements of platform and product compatibility in this document shall be considered individually in relation to the products referred to in the relevant statement, i.e., where any Epicor software is stated to be compatible with one product and also stated to be compatible with another product, it should not be interpreted that such Epicor software is compatible with both of the products running at the same time on the same platform or environment. Additionally platform or product compatibility may require the application of Epicor or third-party updates, patches and/or service packs and Epicor has no responsibility for compatibility issues which may be caused by updates, patches and/or service packs released by third parties after the date of publication of this document.

Table of Contents

About this Manual

The API: Getting Started guide provides you with the information needed to begin using the Prophet 21 API.

This manual includes:

- Roles and responsibilities for the people involved in the API
- An introduction to the API
- How to use the API

This document is not intended to replace the SDK documentation available from the Prophet 21 API home page. The SDK Documentation is the most comprehensive reference for API consumption, and can be found at this URL once your API is deployed*:*

http://[hostname:port]/docs/SDK/webframe.html

> ***Important!*** If you have not installed the API, you will not have access to the SDK documentation.

API Overview

The API is a set of services that enable you to design your own applications that access Prophet 21's business entities and data in real time. For example, if you build an application to create customers, any customers added or updated through that application are immediately available for processing in Prophet 21. Likewise, if you build an application to create orders, orders created through the application are immediately available for processing. You do not have to import data created via the Prophet 21 application.

API is a tool for your own internal Prophet 21 development, or as an integration point for your business partners. As such, the API requires a comprehensive understanding of the Prophet 21 ERP.

What's in the Box?

The API is a website that hosts web services. The IIS server manages the incoming web service calls and relays them to Prophet 21 sessions that, in turn, execute the business logic. The Prophet 21 API installation includes service management software, a non-visual version of the Prophet 21 application, and the related support components for both. The API runs against the Prophet 21 database.

The API expects to run against a version of the Prophet 21 database that precisely matches the non-visual version of the Prophet 21 application, by version number.

For more information about how all of the components of the API interact, please refer to the **API Architecture** topic in the SDK documentation.

API Home Page

The API home page is the primary landing page for the API. This page holds many useful resources, including the following:

- **SDK Documentation** – The SDK documentation provides a wealth of knowledge to help you use the web integrated services integration with Prophet 21.
- **API Console** – Use the API console to create API consumer keys to authenticate the API instead of using user credentials.
- **Release Notes** – Introduced in version 12.16, the Release Notes provide helpful information on deployment modifications, web service additions or behavior changes which impact API consumption specifically.
- **Hosted REST and SOAP services** – The REST and SOAP services lists display a complete list of available web services.
- **API Log Files** – The API log files contain any errors and warnings related to API activity.

Roles

If you are reading this guide, you are a Prophet 21 user who has been asked to do one of the following:

- Install and Maintain the Prophet 21 API server
- Create applications that consume the API and access Prophet 21 business logic and data
- Manage a team of outside consultants who are attempting to use the API
- Assess Prophet 21's API capabilities to determine whether the API is a good fit for your company

To get the most out of your API, users in each role must bring a certain level of expertise. Review each of the following roles to better understand who is responsible for each step in the process.

> ***Important!*** When beginning your integration, each role should work together to identify what you need the API to do, and then determine the best way to leverage the API to satisfy your business needs. Contact Epicor Professional Services for assistance with your implementation needs.

Installation and Maintenance

An IT professional who understands the administration of a web server (IIS), including ongoing maintenance and performance analysis should install and maintain the API server. You may be called on to modify your IIS deployment (both hardware and software configurations) according to your deployment experience and business needs.

For more information on installing and maintaining the API, please refer to the API Installation Instructions found on the customer website.

Developer and Development Manager

Developers and development managers must be familiar with web service consumption, as well as multi-tier application development and deployment. As such, developers and development managers should also be familiar with REST or SOAP services, as well as .NET development.

Developers and development managers need to understand your Prophet 21 business operations to accurately develop the content of the web services calls.

For more information on developing the API, please review the *Using the API* on page 10 section of this document.

Business Analyst

Anyone assessing the API needs a thorough understanding of your company's business needs. Business analysts need to know the capabilities of Prophet 21, as well as their IT and developer resources.

Epicor's Professional Services are available to assist throughout the API development process.

Using the API

Using the API is a matter of calling the web service methods that meet the needs of your own application. It may be as simple as retrieving a customer, or as complex as submitting orders. The API is a powerful tool that accesses the business logic in Prophet 21 on your own terms.

Before developing the API, consider the following:

- Security requirements
- Who will access the web services
- Test system management

Securing the API

You can use the web services out of the box, as deployed by the API website, for internal consumption; however, when exposing services for external partners or third party devices, you should take further security considerations. Employ one or more of the following security strategies:

- Set up a proxy IIS server to deploy your web services, which wrap the internal Prophet 21 services. This tailors your business logic to exactly what your customers need, no more, no less. Use this method to design third party applications or your own applications, create isolation for contracts, and protect them going forward.
- Restrict access to subsections of the API deployment using third party consumer keys. Use this method to limit access to specific branches of the API service tree. While this method is easy to implement, it does not allow you to tailor the look and feel of the web service based on the consumer.
- Implement HTTPS over SSL when required.

For more advanced information on securing the API, please refer to the **Security and Authentication** topic in the SDK documentation.

Web Services

The API web services process domain objects. A domain object represents a logical entity within the Prophet 21 business system. If you are familiar with the database schema, this may translate to tables; however, in some circumstances, a single domain object may represent a grouping of multiple tables. For example, the OE_HDR is the table, but Orders is the related domain object.

Likewise, domain objects may include detailed sub domain object logic alongside or within them. For example, Lines exist within Orders.

To find information about the capabilities of each specific endpoint, click the web service link on the API home page. Doing so provides the following information:

- REST – Callable URLs, verbs, and parameters
- SOAP – WSDL definition

If the service operates on a domain object, you also receive the full domain object definitions.

The API uses three call types:

- **CRUD services** – CRUD services allow for create and update operations on a given domain object. Using CRUD services, you can perform Get operations by an ID or variable criteria. CRUD services accept a domain object as content and return domain objects.
- **Action/Functional services** – Action and functional services perform functionality based on parameters passed into the service. Use these services to perform system actions rather than manipulate domain objects. For example:
  - **GetPrice** – Accepts an item and context information and returns a pricing object.
  - **InventoryMovement** – Creates an inventory adjustment that corresponds with moving inventory from bin to bin. This uses item and bin context information and returns the quantity that Prophet 21 was able to move.
- **Pass through services (specialty calls)** – Pass through services are highly specialized web service calls that require specific functional knowledge. For example, Ecommerce /XBG calls pass specific documents as a parameter with variable result documents. These calls require specialized knowledge of B2B functionality.

Note: Before using pass through services calls, contact Support for related documentation. Locate Ecommerce call documentation in the CodeExamples resource folder. As of version 12.16, find this documentation and XML samples in the Ecommerce folder.

Using Web Services

API service calls are a multi-step process. First, acquire a token that permits the API access to your running session. Each subsequent call against the API must include that token information. Please review the **Code Examples** topics in the SDK documentation for more information on tokens.

> ***Important!*** When exploring the API and submitting calls, it is very easy to change data. Make sure you are configured to call against a play database before experimenting with web service execution and code examples.

The following outlines a typical CRUD web services operation:

Create a New Domain Object

1. **GetNew** – Use this call to set up a domain object template; you do not need to store templates locally to create new data:
   - **GET http://[hostname:port]/api/sales/orders/new**
2. Populate the domain object or XML document with your data. You do not need to provide all XML fields in the domain object; rather, only include the critical fields. Similar to the imports in Prophet 21, transactions require only a small subset of available columns.
3. Update the domain object:
   - **POST http://[hostname:port]/api/sales/orders/**

GET by Query String

Use **GET http://[hostname:port]/api/sales/orders?$query=PoNo EQ 'MYPONO'**

For more information on the possible Query operations, review the query syntax options in the **Query Filters** topic in the SDK documentation.

Use the web service calls as you would any SOAP or REST web service. Alternatively, you can use the provided SOA Rest Client to implement your call in a more traditional programming environment.

The Prophet 21 SOA CodeExamples application, included in every SOA installation, provides an opportunity to execute simple web services (via the .exe) or examine simple code samples (via the .sln) that you can use to incorporate the API into your existing applications.

The following example illustrates how to create an order using the SOA REST Client in a C# application. This example assumes that you have already acquired the ApiToken.

```
RestClientSecurity rcs = RestResourceClientHelper.GetClientSecurity
(ApiToken);


P21.Soa.Service.Rest.Sales.OrderResourceClient orc =
new P21.Soa.Service.Rest.Sales.OrderResourceClient([hostname:port] +
"/api", rcs);


P21.DomainObject.Sales.Order.Order myOrder = orc.Resource.GetNewOrder
();


myOrder.CustomerId = 999999;
myOrder.CompanyId = "MYCOMP";
myOrder.ShipToId = 999991;
```

```
orc.Resource.CreateOrder(myOrder);
```

This is enough data to create an order in Prophet 21. Remember, each domain requires different minimum data sets. For example, you need to specify a CompanyId on most domain objects. Because of this, knowledge of the Prophet 21 system expedites development.

When performing a GetNew, GetByID or GetByCriteria operation, if you want to include sub domain objects in your request, include the **"extendedproperties"** qualifier. The extended property can either be a specific sub domain object or * for all. For example:

- **GET http://[hostname:port]/api/sales/orders/100999?extendedproperties=***
- **GET http://[hostname:port]/api/sales/orders/100999?extendedproperties=Lines**

Refer to the **Data Contracts** topic in the SDK documentation for more information.

You can also use the **extendedproperties** convention when using the REST client. Review the **REST Client Extended Properties Code Example** topic in the SDK documentation for more information.

Best Practices

When working with the API, follow these best practices:

- Allow the system to default any columns in a domain object. When in doubt, opt to not include a column in the domain object. Not populating a given tag or domain object property does not remove the property; rather, the field defaults in the case of creation or remains unchanged in the case of updates.
- When updating existing data, use the appropriate UID and key identifying fields. If you attempt to update a row without using identifying fields (such as a UID, transactional numbers, line numbers, and so on), the system attempts to insert new data instead of performing the update.
- When working with sub domain objects, include the key information for the parent domain object even though this may seem redundant. This is intentional to clarify the programmer's intent. For example, the order number information must be populated through detail and sub domain objects.
- The API is non-visual and does not use interactions to verify user actions. Rather, the system detects key information and acts accordingly. The API throws exceptions and records errors when the system is not able to complete a task. You must submit information as intended by the application. When using the API, account for exceptions and feedback from Prophet 21 and return that information to the user.

Troubleshooting

The API may generate the following errors:

400 Bad Request

Something may be wrong with your data content. Services are very specific when handling content: XML document naming is case sensitive and SOAP is particular about node sequences. If you receive this error, check for case or name errors. Try reducing the XML document to known, valid data, then add a few columns at a time to find the problem. Remember to first use test data before hitting your live database.

Error message in response document

Usually these errors correspond to a business rule failure message in the returned exception. As an API consumer, make sure that you have the ability to let your end users know about exceptions thrown during processing. Review the error logs available from the API home page:

- **P21.SOA.log** – The P21.SOA.log contains a record of service reception and processing from the IIS prospective. Depending on your log settings, you may see incoming content and responses. You may also see information about the way the API brokers incoming jobs to managed mini Prophet 21 instances that handle responding to business logic requests.
- **P21.API.log** – The P21.API.log records Prophet 21 side process business logic responses. Depending on your log settings, you may see process startup and shutdown, initialization and some business logic validation. During your API implementation, if the web responses are not providing detailed enough error messages, review your Prophet 21 logs to see if they provide any additional information that could help you diagnose the error.

Frequently Asked Questions

How do I test my web services?

Use the code examples for Advanced Rest Client, PostMan, and SoapUI.

What you want to do and how much development experience you have determines the tool set you use for exploring the API and your proof of concept work. There are numerous web services harnesses available online.

What's a domain object?

A domain object represents a logical entity within the Prophet 21 business system. Domain objects often translate to tables in the database; however, in some circumstances a single domain object may represent a logical grouping of multiple tables. For example, Oe_hdr is the table, but Orders is the related domain object. Domain objects can include detail domain objects logically organized within them. For example, Lines exist within Orders.

Why don't I see the P21.API.log?

By default, the API does not generate logs. On production servers, logging is disabled to maximise performance. Modify the P21.API.log state on the API home page SOA Configuration utility after the SOA deployment.

Why isn't my document being accepted?

Remember that REST and SOAP endpoints are case sensitive. Double check your naming convention on XML documents or node sequence for SOAP. If these are correct, verify the sub domains within the domain object.

What do I need to submit to get a web service to work?

Check the domain object or parameters associated with the web service call that you are attempting to make. If your service takes a domain object as content, there is a good chance you won't need to provide near the full set of information. For services that accept parameters, required and optional information depends on the individual service.

Is there a list of changes by release?

Yes and no. The API comprises functionality across the entire ERP; it would be impossible to represent every behavioral change between releases. Review point release documentation and lists of errors corrected in the knowledge base.
Version 12.16 introduced Release Notes, available on the API home page. The Release Notes provide

helpful information on deployment modifications, web service additions, and behavior changes which impact that API consumption specifically.

What are my query syntax options?

The API supports numerous query options. For a complete list of query options, review the **Query Filters** topic in the SDK documentation.

I have security concerns.

The API provides both user and consumer authorization via tokens and support https over ssl. Your deployment staff should be comfortable with https and ssl. Epicor provides the tools for security; however, you are responsible for implementing and maintaining your deployment integrity. Contact your IT staff or Epicor Prophet 21 Professional Services for additional assistance. You can also review the **Security and Authentication** topic in the SDK documentation.

Do I have to use XML?

When using SOAP, you must use XML. REST, including the SOA REST client, accepts both XML and JSON. See the code examples for information on setting the Accept Type on the SOA REST client. For direct web service consumption, you can set the Accept Type and Content Type independently.

What about extensibility and my user defined fields (_ud tables)?

User defined fields come through in the API and appear within the domain object most closely associated with the user defined table. User defines fields appear as a subsection of the domain object as the <UserDefinedFields> element. User defined files are subordinate to the domain object to which you they are embedded; if you cannot modify the domain object, you will not be able to modify its user defined fields in the API.

The same requirements for supplying key information that apply to the domain objects also apply to user defined field elements. The presence of key information indicates to the API the _ud table should be updated in the database. The absence of key information indicates to the API the _ud table should be inserted in the database.

In the User Defined Fields section of the API home page, toggle the **User Defined Field Enabled Setting** to **Enabled**. If your user defined fields do not immediately display when GETing domain object data, click **Regenerate User Defined Fields**.