YAROWA

ERP Integration Handbook

Technical Guideline for Craftsman ERP System Integration

October 2025

Introduction

This document describes the REST API for integrating external applications with Yarowa. Yarowa is a SaaS platform for order and vendor management that connects principals with service providers (e.g., craftsmen). By using this API, developers can integrate third-party systems with Yarowa in a standardized, efficient way using RESTful principles and JSON-based communication.

In the core process, a principal creates an order and assigns it to a service provider. The service provider can either accept or decline the order. After the service has been delivered, the process is completed through invoicing: the service provider submits the invoice via Yarowa, which is then forwarded to the principal. The principal may accept or reject the invoice. In case of rejection, the service provider has the opportunity to resubmit a corrected invoice.

Additional optional features of the platform – such as to-do lists, chat messages on order level, partial invoices, or quotation processes – complement the core process but are only covered marginally in this documentation.

This API documentation focuses exclusively on order management. Vendor management functionality is not part of this documentation.

This REST API documentation is specifically designed for the service provider side and enables third-party systems (e.g., ERP or craft business software) to be integrated flexibly. The depth of integration may vary: from simple notifications via WebHooksto full support of actions such as accepting or declining orders, or submitting invoices.

This document is structured to guide developers step by step: first covering core concepts and authentication, followed by detailed REST endpoints, WebHook events, error handling, and optional features.

We provide full support for implementing the interface. API extensions are available upon request. Special requirements from ERP vendors of service providers can also be accommodated (subject to additional charges).
You will find the contact details of our representative for individual requests at the end of this document.

Offers & Expense Cap Increase

- Submit an offer or expense cap increase
- Revise a rejected offer or expense cap increase

Billing

- Submit a partial invoice
- Revise a rejected partial invoice
- Submit a closure / final invoice
- Revise a rejected closure / final invoice

Documents

- Upload Documents
- Get meta-data for attached documents
- Download document content

Chat Messages

- Send a message
- Get all chat messages of a work order
- Get one specific chat message of a work order

Getting Started

This section provides the essential information to start integrating third-party applications with **Yarowa**.

Prerequisites

Before using the API, ensure the following:

- **Yarowa Sandbox Account**
  Software vendors can request a sandbox account from Yarowa. Please contact the representative listed at the end of this document.
- **API Key & Integration Activation**
  You can independently activate the integration and obtain an API key via the Yarowa dashboard:

Integration Management Dashboard
`https://appenzell-de-test.yarowa.io/dashboard/integrationManagement/`

Steps:

1. Select **“Add Integration”**.
2. Choose **“Standard PropertynTrade Gateway”** in the menu.
3. Select the desired authentication method: **Public Key Infrastructure (PKI)** or **JWT Bearer**.
   - **JWT Bearer** is strongly recommended.
4. After activating the gateway, the credentials will be displayed in the Management Dashboard.

Your system must support **HTTPS** requests and **JSON** payloads.
Recommended: a development or sandbox environment for testing.

Sandbox Base URLs

Yarowa provides two environments with separate endpoints:

Shared API
`https://gw-ptde-pool.enable.jarowa.de/shared/`

Use this URL for:

- WebHook registration and event handling
- Chat functionality
- Document management

Property API
`https://gw-ptde-pool.enable.jarowa.de/property/`

Use this URL for:

- Work Order handling (accepting, declining, submitting invoices, etc.)

All requests and responses in the sandbox mirror the production environment in structure and behavior.

Authentication

Yarowa API uses OAuth2.0 authentication.

First Request Example

Here is a simple example to retrieve your orders as a service provider using curl (Property API sandbox):

```
curl -X GET "https://gw-ptde-
pool.enable.jarowa.de/property/api/v1/WorkOrder" \
     -H "Authorization: Bearer YOUR_AUTH_TOKEN" \
     -H "Accept: application/json"
```

Expected Response (HTTP 200):

```
[
  {
    "WorkOrderId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "ServiceInfo": {...},
    "PrincipalInfo": {...},
    "ProviderInfo": {...},
    "Main": {
      "status": "placed",
      "caseId": "23456",
      "orderId": "38838339",
      [...]
    }
  },
  ...
]
```

Sandbox / Test Environment

All requests in the sandbox are safe to test and do not affect live production data. Sandbox credentials are obtained separately from your API key. The sandbox mirrors the production environment in structure and behavior.

Next Steps

After verifying your first request:

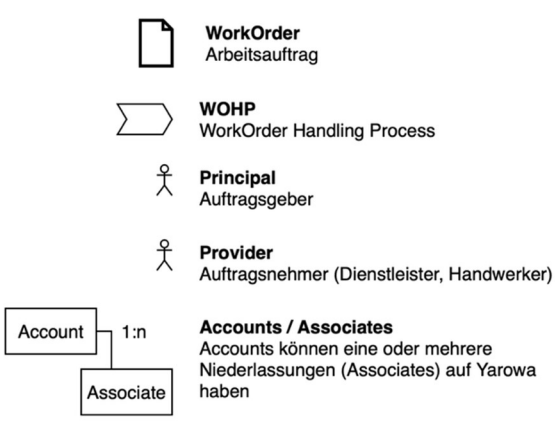
- Familiarize yourself with **core objects and order lifecycle**.
- Explore available **REST endpoints** for Work Order handling and other features.
- Set up **WebHooks** using the Shared API to receive real-time event notifications.

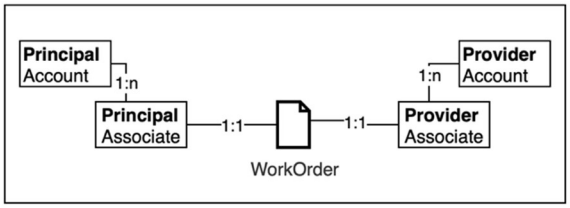
Core Concepts

This section describes the main objects and concepts in Yarowa relevant for service providers using the API.

Key Objects

Object	Description
Work Order	Represents a work order created by a **principal** and assigned to a **service provider**. Orders include details such as scope of work, deadlines, and attachments.
Invoice	Represents a billing document submitted by the **service provider** after completing the order. Invoices can be accepted or rejected by the **principal**.
Principal	The client or organization that creates and assigns orders to service providers.
Service Provider	The contractor or company that receives and executes orders from principals.
Associate	An associate can be, for example, a department within the principal's company or several branches of a service provider.
WebHook	Mechanism to notify external systems about events such as order creation, acceptance, rejection, or invoice submission.
Chat / Documents	Optional objects linked to an order, e.g., messages, to-do items, or supporting documents. Managed via the Shared API.

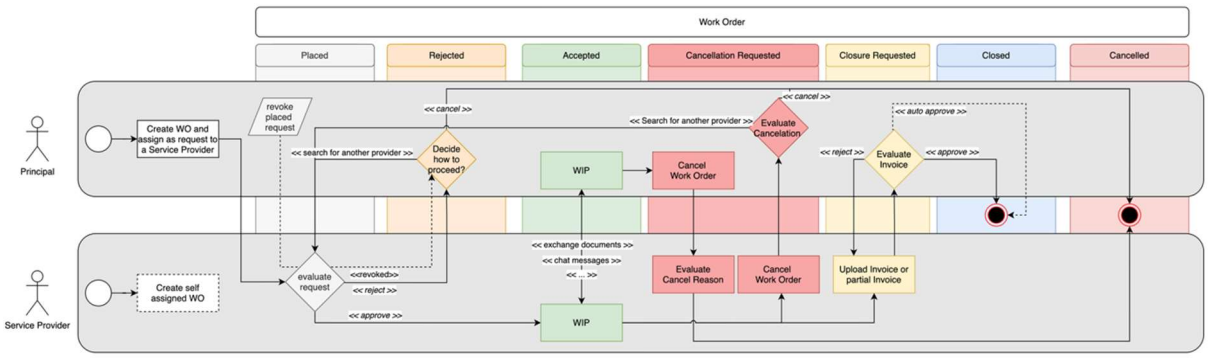




In Yarowa, a **principal** creates **work orders** and assigns them to **service providers**. Both principals and service providers can have one or more **associates**, and each work order is always established between a specific pair of associates. The Work Order Handling Process (**WOHP**) defines how the work order progresses and how data is exchanged between the associates, following a classical state machine model that ensures each step of the workflow is tracked and managed systematically. With each state change of a work order, an **event** is triggered and send to registered **WebHooks**.

Order Lifecycle

The typical lifecycle of an order:



1. **Created** – The principal creates an order and assigns it to a service provider.
2. **Placed** – The service provider sees the order in the system and decides to **accept** or **decline** it.
3. **Accepted** – The service provider has confirmed the order and starts working on it.
4. **Rejected** – The service provider has rejected the order. The principal may reassign it to another provider or cancel it.
5. **Cancellation Requested** – The service provider or principal has decided to cancel an accepted work order.
6. **Cancelled** - The cancellation request was confirmed by the other party or a rejected work order was cancelled by the principal.
7. **Closure Requested** – After completing the work, the service provider submits an invoice via the Property API.
8. **Invoice Accepted / Closed** – The principal accepts the invoice, completing the order.
9. **Invoice Rejected** – The principal rejects the invoice. The service provider may correct and resubmit it.

Each stage triggers events via **WebHooks**, allowing external systems to stay synchronized with the order status in real time.

Integration scenarios

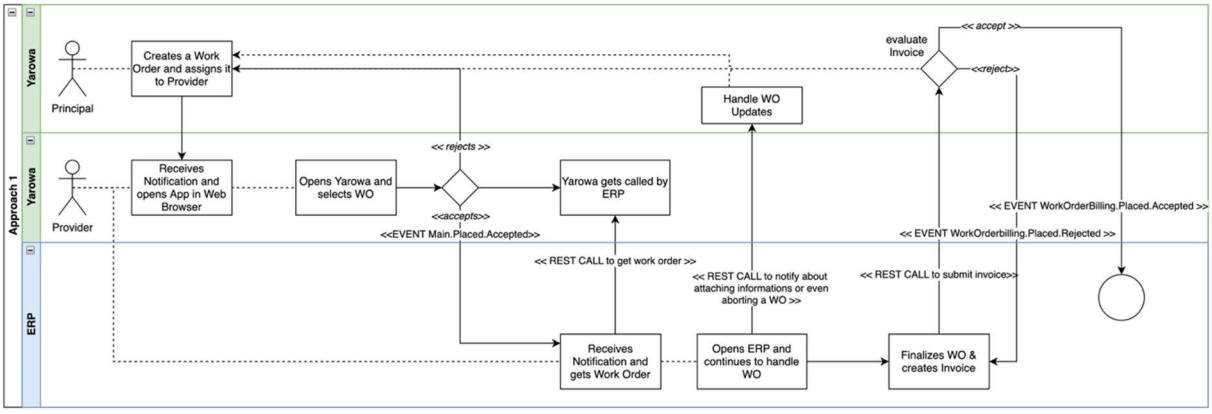
This section illustrates two typical approaches for integrating third-party systems with Yarowa. It highlights the flexibility of the platform, ranging from minimal integration to full integration, depending on the requirements and capabilities of the service provider's system.

Scenario 1 – Minimal Integration

In this scenario, the service provider primarily works on the Yarowa platform. The third-party system is used mainly for internal tracking or reporting.

Process Flow:

1. A **principal** creates a work order in Yarowa.
2. The **service provider** accepts the order directly on the Yarowa platform.
3. The service provider's third-party system retrieves accepted orders from Yarowa, either automatically or on demand via the Yarowa API**.**
4. The order data is imported into the third-party system for internal use.
5. The service provider completes the work in the third-party system and generates an invoice.
6. The invoice can be submitted to Yarowa via the API or exported from the third-party system and manually uploaded to Yarowa.
7. Optional features such as **chat, document management, or quotation processes** are used directly on the Yarowa platform and are not synchronized with the third-party system.



Key Characteristics
Low implementation effort.

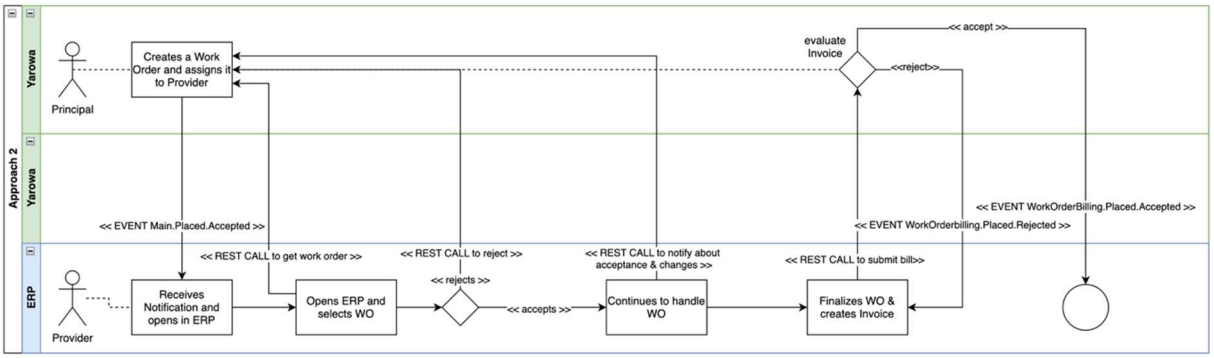
- Yarowa remains the primary platform for work order management and optional features.
- The third-party system is mainly used for internal tracking, reporting, or invoicing.

Scenario 2 – Full Integration

In this scenario, the service provider's third-party system fully replaces Yarowa for day-to-day work order handling. The integration enables the service provider to perform all actions, including accepting, declining, and invoicing, directly within their own system.

Process Flow:

1. A **principal** creates a work order in Yarowa.
2. The work order is automatically pushed to the service provider's third-party system via the Yarowa API.
3. The service provider receives the order in their system and can immediately accept or decline it.
4. All subsequent actions, including task updates, document uploads, chat messages, and invoicing, are performed entirely within the third-party system.
5. All changes are synchronized back to Yarowa in real time via the API, ensuring that the principal always sees the current status of the order.



Key Characteristics:

- High level of automation, with minimal interaction on the Yarowa platform.
- Enables full workflow handling within the third-party system.
- All events and updates are synchronized in real time, including invoices, status changes, and optional features like chat or documents if supported.

WebHooks & Event Handling

Yarowa supports **WebHooks** to notify third-party systems about important events in real time. This chapter explains how to register WebHooks, handle events, and manage retries in case of errors. For a detailed endpoint description, please see chapter "REST Endpoints".

Registering a WebHook

To receive events from Yarowa, you must register a WebHook endpoint in your system.

Steps to register a WebHook:

1. Choose an endpoint in your system that will receive POST requests from Yarowa.
2. Use the endpoint `/api/v1/SetConfig` in the **Shared API** to create the WebHook by sending a POST request to the WebHook registration endpoint.
3. Include the following information in the request:
    - `url`: Your endpoint URL
    - `events`: List of events you want to subscribe to
    - `secret`: Optional secret key for request signature validation

Handling Events

When Yarowa sends an event to your WebHook endpoint:

- If your system **returns HTTP 200 OK**, the event is considered successfully delivered, and no retry will occur.
- If your system returns a **non-200 status** (e.g., 500 Internal Server Error), Yarowa will retry sending the event according to the retry mechanism described below.

Retry Mechanism

Yarowa retries failed WebHook deliveries automatically:

- Retry occurs with max(1, RetryCount)^3 backoff (after 1 min, 8 min, 27 min, etc.)
- Retries continue until the event is successfully delivered or the maximum retry limit of 10 is reached – includign the initial request.
- If the events can't be published sucessfully after 10th retry, the WebHook is marked as blocked and a manual re-send has to be triggered.

To manually re-send events in case of persistent issues, you can use the **event replay API** in the Shared API, which allows resending events to the registered WebHook endpoint.

Event Naming Convention

All WebHook events follow a structured naming convention:

`<Flow>.<FromStatus>.<ToStatus>`

- **Flow**: The process or sub-flow in which the event occurs.
- **FromStatus**: The previous status before the change.
- **ToStatus**: The new status after the change.

Flows	Description
`Main`	Main workflow for Work Orders
`AppointmentProcess`	Sub-flow for scheduling appointments
`ExpenseCapIncrease`	Sub-flow for expense limit changes
`WorkOrderBilling`	Sub-flow for invoice and billing events
`WorkOrderFileUpload`	Sub-flow for file attachment events
`Conversation`	Sub-flow for chat or messaging events

Example

- `Main.Created.Placed`
  The main workflow transitioned from `Created` to `Placed`.

- `WorkOrderBilling.Created.Placed`
  An invoice was submitted and its status changed to `Placed`.

Events

Events are triggered as soon as there are work order changes on Yarowa Plattform. For some actions, more than one Event might be triggered. This table includes the most relevant events, additional events may exist depending on enabled features.

Event	Description
Work Order Creation / Reassign / Cancellation / Rejection	
`Main.Created.Placed`	A work order has been created and placed by a principal.
`Main.Recalled.Placed`	A work order was taken back from first provider and reassigned.
`Main.Rejected.Placed`	A rejected work order is reassigned to another provider
`Main.Placed.Accepted`	A work order is accepted by the provider.
`Main.Accepted.CancellationRequested`	An accepted work order is cancelled.
`Main.CancellationRequested.Cancelled`	The other party has accepted the cancellation request-
`Main.Accepted.ClosureStarted`	The provider started the final invoice process but hasn't sent an invoice to the principal yet.
`Main.ClosureStarted.ClosureRequested`	The provider has sent an invoice to the principal. *See Billing flow description above for further events.*
`Main.ClosureRequested.Closed`	The principal has approved the invoice and the work order is closed.
Appointments	
`AppointmentProcess.Created.Done`	An appointment was set or rescheduled
`AppointmentProcess.Draft.`	A set appointment was cancelled
ExpenseCap increase and offers	
`ExpenseCapIncrease.Created.Placed`	An offer is submitted
`ExpenseCapIncrease.Placed.Rejected`	An offer is rejected and a new revised offer is requested.
`ExpenseCapIncrease.Rejected.Placed`	An revised offer is submitted
`ExpenseCapIncrease.Placed.Accepted`	The offer is accepted
`Main.Accepted.Cancelled`	The offer is rejected and the work order is cancelled.
Billing flow	
`WorkOrderBilling.Created.Placed`	A bill is submitted to the principal.
`WorkOrderBilling.Placed.Rejected`	The bill is rejected.
`WorkOrderBilling.Rejected.Placed`	A revised bill is submitted.
`WorkOrderBilling.Placed.Accepted`	A bill is accepted
Document Upload	
`WorkOrderFileUpload.Draft.`	A document is uploaded and attached to a work order
Chat Message	
`Conversation.Created.Placed`	A new chat message was posted.

Best Practices

- Always respond with **200 OK** as quickly as possible to avoid unnecessary retries.
- Validate event payloads using the secret key if provided.
- Log received events for auditing and debugging.
- Use idempotent handling to avoid duplicate processing in case of retries.

REST Endpoints

In this chapter the most important endpoints for the service provider side third-party systems (e.g., ERP or craft business software) are listed.

The purpose of this chapter is to describe the functionality of each endpoint. Swagger is, however, the first source of API-Documentation.

SHARED-API https://gw-ptde-pool.enable.jarowa.de/shared/swagger/index.html
You can find a detailed Swagger API documentation at the link above for the WebHook registration and event handling, chat functionality and Document management.

PROPERTY-API https://gw-ptde-pool.enable.jarowa.de/property/swagger/index.html
Contains all endpoints for the Work Order handling (accepting, declining, submitting invoices, etc.). For the implementation for service providers only the endpoints in the swagger sections “ProviderUser” and “WorkOrder” are relevant. The others can be ignored.

Web Hooks

Setup Web Hook

SHARED-API

POST /api/v1/SetConfig

Condition: none

This endpoint can be used to set how you want to receive the WorkOrder changes Events and any relevant authentication/authorization. Webhook: This setting sends the Events to your Webhook endpoint. You can set your Webhook URI and authentication method of your Webhook endpoint yourself via webhookSettings object of SetConfig endpoint.

Currently, it only supports OAuthClientCredentials for your Webhook-endpoint authentication. At your request, wecan extend this setting with another authentication method.

The object eventFilter of SetConfig endpoint can be used to whitelist or blacklist certain events that the customer does or does not want to receive. The eventFilter part can also be omitted, which essentially disables the filter, so all events will be sent.

Get pending events

SHARED-API

GET /api/v1/queue/GetPendingEvents

Condition: none

This endpoint is used to check if there are any pending events that have not been successfully sent to your webhook endpoint or have not been stored in the Azure storage table to be retrieved by the Get Event endpoint.

Trigger a retry for pending events

SHARED-API

POST /api/v1/queue/RetryPendingEvent

Condition: none

This endpoint is used to resend the pending events if there are any. Our Gateways are designed to make up to 10 automatic retry attempts for sending pending Events, after which the events are flagged as blocked or stuck. To reset this counter, you can use the POST RetryPendingEvent endpoint.

Work Orders

The Work orders API is the collection of all endpoints that mainly process the work orders created by a principal and assigned to a service provider. The service provider can decide to accept or reject these work orders, receive work order details and to cancel already accepted work orders.

Get work order details

PROPERTY-API

GET /api/v1/WorkOrder

Condition: none

This endpoint is used to get the complete details (all attributes) present in the given Workorder. This endpoint does not return the Chat Messages or the Documents that can be retrieved via other available endpoints.

Yarowa processes different types of work orders, as defined by the field requestedDelivery.

A work order of type “repair” is the standard work order without any limitations.

A “repair_withExpenseCap” indicates a work order that has a predefined expense cap, which must not be exceeded. This cap is part of the work order details.

A “repairWithServicePositions” refers to a work order that includes a list of service positions. The service positions are an array of table rows detailing the exact work that needs to be performed.

Lastly, the work order type “offer” is used when an offer must be sent to the principal before any work is carried out. Please refer to the endpoints in Chapter “

Offers & Expense Cap Increase” for instructions on how to handle offers.

- No triggered Events
- No status change

Accept or reject incoming work orders

PROPERTY-API

Work orders that are assigned to service providers can be accepted or rejected by using the follow endpoints.

POST /api/v1/actions/ProviderUser/AcceptWorkOrder

Condition: Status Placed

This endpoint is used to accept the WorkOrder assigned to you by the Principal. You can retrieve all the details of the WorkOrder via GET WorkOrder endpoint before accepting/rejecting a WorkOrder.

- Triggered Events
  - Main.Placed.Accepted
- Status changes to: Accepted

POST /api/v1/actions/ProviderUser/RejectWorkOrder

Condition: Status Placed

This endpoint is used to reject the WorkOrder assigned to you by the Principal. You can retrieve all the details of the WorkOrder via GET WorkOrder endpoint before accepting/rejecting a WorkOrder.

- Triggered Events
  - Main.Placed.Rejected
- Status changes to: Rejected

Cancel or confirm cancellation

PROPERTY-API

If a work order was accepted by the service provider, it can be cancelled by both parties. If it was cancelled by the principal, then a confirmation of the cancellation is required. You can perform the two actions by the following endpoints.

POST /api/v1/actions/ProviderUser/CancelWorkOrder

Condition: Status Accepted

This endpoint is used to cancel the WorkOrder assigned to you by the Principal which you have already accepted. This endpoint can only be executed in Workorder-status Accepted.

- Triggered Events
  - Main.Accepted.CancellationRequested
- Status changes to: CancellationRequested

When the Principal accepts the cancellation, then the work order changes to status Cancelled.

POST /api/v1/actions/ProviderUser/AcceptCancelRequest

Condition: Status CancellationRequested

This endpoint is used to confirm a cancellation of the WorkOrder, cancelled by the Principal.

- Triggered Events
  - Main.CancellationRequested.Cancelled
- Status changes to: Cancelled

Appointments

Service providers can use the appointment notification endpoints to share appointment details related to a work order. When an appointment is submitted, the **main contact person** of the work order receives an SMS and/or email with the provided information.

Set an appointment

PROPERTY-API

POST /api/v1/actions/ProviderUser/SetAppointment

Condition: Status Accepted

This endpoint is used to store an appointment in a WorkOrder that has been agreed with the end customer.

- Triggered Events
  - AppointmentProcess.Created.Done
- No status change

Reschedule an appointment

PROPERTY-API

POST /api/v1/actions/ProviderUser/RescheduleAppointment

Condition: Status Accepted and an appointment was already set

This endpoint is used to store a rescheduled appointment in a WorkOrder that has been agreed with the end customer. This endpoint can only be executed if the appointment is already stored in a WorkOrder.

- Triggered Events
  - AppointmentProcess.Created.Done
- No status change

Cancel and abandon appointment

PROPERTY-API

POST /api/v1/actions/ProviderUser/CancelAppointment

Condition: Status Accepted and an appointment was already set

This endpoint is used to cancel an appointment stored in a WorkOrder. This endpoint can only be executed if the appointment is already stored in a WorkOrder.

- Triggered Events
  - AppointmentProcess.Draft.
- No status change

POST /api/v1/actions/ProviderUser/AbandonAppointment

Condition: Status Accepted and previous appointment was cancelled

This endpoint is used to abandon an appointment flow in case no appointment is required to be stored in a WorkOrder. This endpoint can only be executed either before storing any appointment in a WorkOrder or after cancelling the stored appointment with CancelAppointment endpoint.

- No triggered Events
- No status change

Offers & Expense Cap Increase

When a work order of type “offer” is assigned to a service provider, the provider is required to submit an offer using the Offer API endpoints. If the principal rejects the offer, the service provider may submit a revised offer through the same endpoints.

Additionally, the Offer API can be used to request an expense cap increase for work orders that are subject to a predefined cost limit. Such requests are sent to the principal for review and approval.

Submit an offer or expense cap increase

PROPERTY-API

POST /api/v1/actions/ProviderUser/CreateOffer

Condition: Status Accepted

This endpoint is used to send an offer or expense cap to the Principal. This endpoint can only be executed in Workorder-status Accepted.

- Triggered Events
  - ExpenseCapIncrease.Created.Placed
- No status change

Revise a rejected offer or expense cap increase

PROPERTY-API

POST /api/v1/actions/ProviderUser/ReviseOffer

Condition: Status Accepted and a previously sent offer was rejected

This endpoint is used to send a revised offer or ExpenseCap to the Principal if he rejects the previously sent offer and demands a revised offer. This endpoint can only be executed when the previous offer was rejected by the Principal and he demanded a new offer.

- Triggered Events
  - ExpenseCapIncrease.Rejected.Placed
- No status change

Billing

The Billing API allows service providers to submit invoices related to a work order. Both partial invoices and a final invoice are supported.

Partial invoices: Service providers can submit partial invoices at any time. If a partial invoice is rejected by the principal, it may be corrected and resubmitted through the API.

Final invoice: When the service provider submits a final invoice, the work order enters the closing process.

If the final invoice is rejected, the service provider can submit a corrected version. If the final invoice is accepted, the work order status changes to Closed.

This process ensures that principals maintain control over invoice approval while service providers can efficiently resubmit corrected invoices when required.

Submit a partial invoice

PROPERTY-API

POST /api/v1/actions/ProviderUser/CreatePartialBill

Condition: Status Accepted

This endpoint is used to send a partial bill to the Principal. This endpoint can only be executed in Workorder-status Accepted.

- Triggered Events
  - WorkOrderBilling.Created.Placed
- No status change

Revise a rejected partial invoice

PROPERTY-API

POST /api/v1/actions/ProviderUser/RevisePartialBill

Condition: Status Accepted and a previously sent partial invoice was rejected

This endpoint is used to send a revised partial bill to the Principal when he rejects the previously sent partial bill. This endpoint can only be executed when the previous partial bill was rejected by the Principal.

- Triggered Events
  - WorkOrderBilling.Rejected.Placed
- No status change

Submit a closure / final invoice

PROPERTY-API

POST /api/v1/actions/ProviderUser/CreateClosureBill

Condition: Status Accepted

This endpoint is used to send a closure bill to the Principal. This endpoint can only be executed in Workorder-status Accepted.

- Triggered Events
  - Main.Accepted.ClosureStarted
  - Main.ClosureStarted.ClosureRequested
  - WorkOrderBilling.Created.Placed
- Status changes to: Closure Requested

When the Principal accepts the final invoice, then the work order changes to status Closed.

Revise a rejected closure / final invoice

PROPERTY-API

POST /api/v1/actions/ProviderUser/ReviseClosureBill

Condition: Status Closure Requested and a previously sent final invoice was rejected

This endpoint is used to send a revised closure bill to the Principal when he rejects the previously created closure bill. This endpoint can only be executed when the previous closure bill was rejected by the Principal.

- Triggered Events
  - WorkOrderBilling.Rejected.Placed
- No status change

When the Principal accepts the revised invoice, then the work order changes to status Closed.

Documents

The Documents API enables service providers to manage files related to a work order. Documents can be uploaded at any time (e.g., invoices, supporting documents, or photos of damages) and are stored as part of the work order. Documents uploaded by the principal can also be accessed, with metadata available for retrieval and file content available for download.

This functionality ensures that all relevant information—such as attachments for invoicing or documentation for the work order—is centrally available to both principals and service providers.

Upload Documents

PROPERTY-API

POST /api/v1/actions/ProviderUser/UploadFile

Condition: none

This endpoint is used to upload a file and attach it to the work order. This endpoint can be used at any time after a work order was accepted and before it is closed or cancelled.

- Triggered Events
  - WorkOrderFileUpload.Draft.
- No status change

Get meta-data for attached documents

SHARED-API

GET /api/v1/{wohpId}/DocumentsMetaData

Condition: none

This endpoint is used to get the metadata of all the documents present in the given Workorder. From this endpoint, you can also get the documentId which can be used to download/retrieve that specific document from the endpoint GET DocumentDownload.

- No triggered Events
- No status change

Download document content

SHARED-API

GET /api/v1/{wohpId}/DocumentDownload/{documentId}

Condition: none

This endpoint is used to retrieve/download a specific document present in the given Workorder based on its id.

- No triggered Events
- No status change

Chat Messages

Service providers and Principals can exchange chat messages related to work orders for example to claify questions related to the work to be done. It is possible to send and receive these conversation messages by the following endpoints.

Send a message

SHARED-API

POST /api/v1/{wohpId}/SendMessage

Condition: none

This endpoint can be used to send chat messages to the other party (Principal/Provider) within a specific WorkOrder.

- Triggered Events
  - Conversation.Created.Placed
- No status change

Get all chat messages of a work order

SHARED-API

GET /api/v1/{wohpId}/GetConversation

Condition: none

This endpoint is used to retrieve all the chat messages within a specific WorkOrder.

- No triggered Events
- No status change

Get one specific chat message of a work order

SHARED-API

GET /api/v1/{wohpId}/GetConversationMessage/{messageId}

Condition: none

This endpoint is used to retrieve a specific chat message within a specific WorkOrder based on its messageId that can be retrieved from GetConversation endpoint

- No triggered Events
- No status change

Yarowa– Your technologically leading partner for simple and fully automated serviceprovider and order management in the Property Management.
Get in touch with us!

Your Contact Person:

Frederic Schoeller

Head of Business Development
Yarowa GmbH

Email	frederic.schoeller@yarowa.com
Phone	+ 49 151 68559959
Home Page	www.yarowa.com
Address	**Yarowa GmbH**, Sonnenstrasse 32 80331 München Deutschland