

# Standard Palletizing Interface

## Revision history

Version	Date	Author	Description
1.0	06-02-2023	AEF SML PPO	Initial version.
1.1	24-05-2023	PPO	Changes to better support PLCs id as strings dashes changed to underscore in JSON keys Added telegrams " Item/container at Location", " Item/container transport request" and "Item/container transport reply"
1.2	11-10-2023	AEF PPO SML	Added sequence numbers and acknowledgements. Added more detailed error handling & reporting. Added palletizing strategies. Updated several telegrams.
2.0	08-11-2023	AEF, SML	Expanded explanation of acknowledgements. Palletizing cells can request layer patterns through info-requests. Several corrections throughout the document. Split up the document into two parts: Standard Equipment Interface & Standard Palletizing Interface.
2.1	12-06-2024	AEF	Updated the equipment interface specification to use from v2.1 to v2.2. Added section for common data types (5.1) Changed how layer patterns are configured (see section 7.1). Updated topic for 6.4.1 Item/Container at Location.
2.2	09-07-2024	AEF SML	Added Change Completed telegram Updated Set Palletizing Telegram Updated state change sequence for container transports Updated examples
2.3	13-03-2025	AEF	Updated to version 2.4 of the standard equipment interface. Added reference to container transport document. Removed redundant telegrams.
	24-06-2025	PDA	Release

## Review history

Version	Date	Reviewer	Notes / description
1.0	03-02-2023	PPO	Review
1.1	09-06-2023	AEF	Minor adjustments to some types and examples (use strings for all ID's).
1.2	11-10-2023	AEF PPO SML FRS	Review
2.0	09-11-2023	SML AEF	Review
2.1	13-06-2024	MIV	Review
2.2	09-07-2024	AEF SML	Review
2.3	13-03-2025 24-06-2025	SML PDA	Review Release

# Table of Contents

<b>1 Introduction</b>	<b>5</b>
1.1 References	5
1.2 Glossary	5
1.3 Limitations of Current Version of Interface Specification	5
1.4 Copyright and right to use under Apache License 2.0	6
1.5 Background / 5G-Robot	6
<b>2 Domain Model</b>	<b>7</b>
<b>3 Palletizing strategies</b>	<b>8</b>
3.1 Clean Layer Pattern Strategy	8
3.2 Mixed Layer/Pallet Strategy	8
3.2.1 Ahead of Time Knowledge	8
3.2.2 Building and Maintaining Digital Twin of Pallet to Build	8
3.2.3 Temporary Layaway Buffers	8
<b>4 Communication</b>	<b>10</b>
4.1 Using the MQTT Protocol	10
<b>5 Common Data Types</b>	<b>11</b>
5.1 Layer Pattern (LayerPattern)	11
<b>6 Error Types</b>	<b>12</b>
<b>7 Common Telegrams</b>	<b>13</b>
7.1 Info-request Telegram (HLC <-> Cell)	13
7.2 Change Completed (Cell -> HLC)	13
7.3 Container Updates and Management	14
7.3.1 Container Placed on Pallet (Cell -> HLC)	14
<b>8 Clean Layer Pattern Telegrams</b>	<b>16</b>
8.1 Configuration	16
8.1.1 Get Layer Patterns (Cell <-> HLC)	16
8.1.2 Add/Update Layer Pattern (HLC -> Cell)	17
8.1.3 Remove Layer Pattern (HLC -> Cell)	18
8.2 Palletizing Actions and Updates	19
8.2.1 Set Palletizing Job (HLC -> Cell)	19
<b>9 Scenarios and Example Flows</b>	<b>21</b>
9.1 Palletizing a Layer	21
9.2 Palletizing a Partial Layer	22
<b>10 Appendix</b>	<b>24</b>
10.1 Layer Pattern	24
10.1.1 The Pallet	24
10.1.2 Describing a Layer Pattern	25
10.1.3 Using a Layer Pattern	26

# 1 Introduction

This document is a specialization of the standard equipment interface [1] that describes the standard palletizing interface. This interface is used for communication between a high-level control (HLC) system and a palletizing cell.

Detailed descriptions of the communication protocol, telegram format, and common telegrams used by this interface can be read in the standard equipment interface document [1]. Readers are advised to become familiar with that document before implementing the standard palletizing interface. Furthermore, this interface uses the container transport handling described in [2], if supported by the palletizing cell.

## 1.1 References

ID	Document	Description
[1]	Stanard Equipment Interface v2.4	The base interface description used for all equipment / robot cell open interfaces. Found on Intelligent Systems website: <a href="https://www.intelligentsystems.dk/products-keep-customers-at-forefront-of-technology/">https://www.intelligentsystems.dk/products-keep-customers-at-forefront-of-technology/</a>
[2]	Standard Transport Interface v0.1	Specification for moving items or containers between a robot cell and external equipment/locations. ound on Intelligent Systems website: <a href="https://www.intelligentsystems.dk/products-keep-customers-at-forefront-of-technology/">https://www.intelligentsystems.dk/products-keep-customers-at-forefront-of-technology/</a>
[3]	Domain Model v1.3	Defines various terms and concepts used for the domain. ound on Intelligent Systems website: <a href="https://www.intelligentsystems.dk/products-keep-customers-at-forefront-of-technology/">https://www.intelligentsystems.dk/products-keep-customers-at-forefront-of-technology/</a>
[4]	MQTT Protocol for Standard Interfaces v1.0	Protocol specification for using MQTT with the standard open interfaces. ound on Intelligent Systems website: <a href="https://www.intelligentsystems.dk/products-keep-customers-at-forefront-of-technology/">https://www.intelligentsystems.dk/products-keep-customers-at-forefront-of-technology/</a>

## 1.2 Glossary

Abbreviation	Description
Cell	Palletizing Cell - robot controller or PLC controlling the robot cell
HLC	<b>H</b> igh- <b>L</b> evel <b>C</b> ontrol system - e.g. WCS/WMS, MES or MFS.

## 1.3 Limitations of Current Version of Interface Specification

The current version of the interface has following limitations which may be addressed in future versions:

- Containers in a layer must have the same height
- Slip sheets can only be placed under a layer - if one is required on the top-most layer, this must be added manually.
- Labeling pallets are not supported.

- Using in-between pallets is not supported.

## 1.4 Copyright and right to use under Apache License 2.0

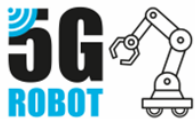
The copyright on this document and any contained specifications, designs, etc. belongs to the company: Intelligent Systems A/S, Havnevej 11, DK-9560 Hadsund, DENMARK.

Any referenced 3rd party designs, technologies and or other IP shall remain with the original owners.

The reference architecture, designs and the included open standard integration interfaces are open and free to use under Apache License 2.0. For details please see

<https://www.apache.org/licenses/LICENSE-2.0>.

## 1.5 Background / 5G-Robot



Parts of this document / release was made in the **5G-Robot** project also known under the long name **5G-ENABLED AUTONOMOUS MOBILE ROBOTIC SYSTEMS** - the largest innovation project that has been launched under the Innovation Fund Denmark's (IFD) Grand Solutions program.

The groundbreaking project united Denmark's leading robot, automation and factory digitalization companies as technology vendors, research partners and industry-leading end-user companies.



Illustration: Project partner logos.

The aim of the project was to revolutionize manufacturing - paving the way to smart production and smart factories and the application of a number of new technologies in production and manufacturing including 5G wireless communication, cloud and edge computing and digital twin.

Intelligent Systems played a leading role in the project, providing the glue that ties the robotic solutions of the partners together making the work as one - i.e. one connected integrated intelligent manufacturing system.

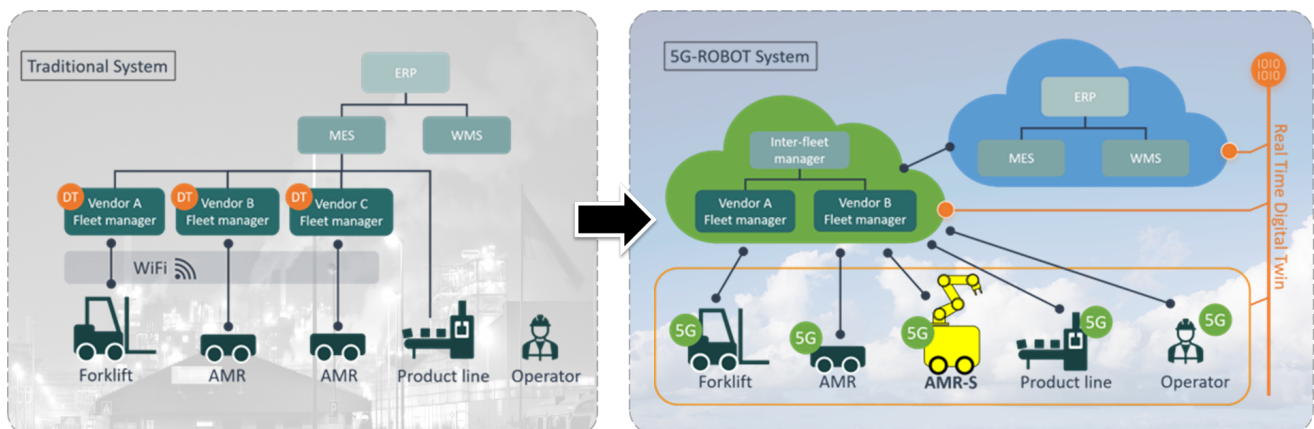


Illustration: The aim is to revolutionize manufacturing paving the way to smart production and smart factories.

Read more about the 5G-Robot project here: <https://www.5gsmartproduction.aau.dk/5g-robot>

## 2 Domain Model

It is recommended to familiarize with the *container*, *palletizing cell* & *palletizing* concepts in the domain model [3].

## 3 Palletizing strategies

The interface currently supports one palletizing strategy.

In a future iteration, the interface will support multiple palletizing strategies. The Clean Layer Pattern strategy and Mixed Pallet strategy are described in detail below.

### 3.1 Clean Layer Pattern Strategy

This strategy is the approach to palletizing boxes of homogeneous sizes of containers onto a pallet and letting the palletizing cell carry out the placement of containers according to the provided layer pattern.

The clean layer pattern strategy enables the creation of pallets with a high degree of stability, with only a single type of containers or pallets with multiple types of containers, but only one type per layer.

### 3.2 Mixed Layer/Pallet Strategy

**Note: This section is not complete and currently only intended for providing considerations when implementing the current version of the interface. It will be supported in a future release.**

In this strategy, the HLC takes full control of where to place a container on the pallet. This is used when building pallets of varying sizes of containers/boxes.

Initial preconditions for containers used the first version of the mixed pallet strategy:

- Are structurally robust and can form a stable pallet
- A number of smaller palletized containers will take up the same amount of space as the one size larger container in the collection. We're thinking along the lines of Lego bricks.

HLC consider the following options:

- Ahead of time knowledge of containers arriving
- Building and maintaining digital twin of pallet to build
- Temporary layaway to a buffer location

#### 3.2.1 Ahead of Time Knowledge

The longer the HLC is able to look ahead as to which containers will be arriving and in which order, the better HLC is able to optimize the layout of the individual pallet. A shorter lookahead is anticipated to cause more holes in the built pallet, potentially reducing the pallet stability.

#### 3.2.2 Building and Maintaining Digital Twin of Pallet to Build

HLC maintains a digital twin of the pallet being built and uses this information to inform the cell exactly where to put the next container on the pallet or whether to use layaway buffer capacity to save the container for later placement on the pallet.

The HLC may, at its discretion, choose to rebuild a smaller part of the pallet based on obtained knowledge about containers which have arrived or are arriving in the short term.

#### 3.2.3 Temporary Layaway Buffers

To improve pallet stability and utilization of capacity we're anticipating that using a temporary layaway of containers for later placement on the pallet will improve the structural integrity of the built pallet as well as utilization of pallet capacity.

A temporary layaway could be a secondary load carrier/pallet which the cell can use for temporary storage of containers.

It should be considered whether the temporary layaway could be a shared resource between two palletizing stations.

## 4 Communication

The telegram format for the interface can be read in the standard equipment interface document [1].

### 4.1 Using the MQTT Protocol

Refer to [4] to see details for implementing the interface using the MQTT protocol.

Note that this interface overrides the MQTT protocol with the following values:

- Prefix is set to *palletizing-cells*
- Publisher is set to *cell* for equipment and *hlc* for the HLC

Example topic for [Setting a palletizing layer](#): *palletizing-cells/cell-3821/cell/palletize/layer*

## 5 Common Data Types

### 5.1 Layer Pattern (LayerPattern)

Data type: **LayerPattern**

The configuration for a layer pattern.

Property	Type	Description
id	String	ID of the layer pattern.
layer_name	String	Name of the layer pattern. Used to improve readability.
pallet_type	String	Type of pallet this pattern applies to.
use_slip_sheet	Boolean	Whether to add a slip sheet before placing containers in the layer.
container_length	Integer	Length of the container.
container_width	Integer	Width of the container.
container_height	Integer	Height of the container.
index	Integer	Index of the container. Starts at 0.
x	Integer	X-coordinate for the top-left corner of the container, given in millimeters.
y	Integer	Y-coordinate for the top-left corner of the container, given in millimeters.
r	Decimal	Rotation of the container, given in degrees.

```
{
  "id": "1",
  "layer_name": "1 box",
  "pallet_type": "EURO",
  "use_slip_sheet": false,
  "container_length": 360,
  "container_width": 280,
  "container_height": 100,
  "positions": [
    {
      "index": 0,
      "x": 0,
      "y": 0,
      "r": 0.0
    }
  ]
}
```

## 6 Error Types

This section extends the error types presented in the standard equipment interface document [1].

The following list of errors are the ones referenced in this interface. The relevant Error IDs are listed for each telegram of which it is relevant.

Error ID	Error Message	Reason
FRAGMENT_MISSING	Missed fragment	<p>One of the fragments were not received, or fragments were not received in the expected order. When the error message is received, the fragments should be sent again, starting at number 1.</p> <p>The expected fragment number that was missing should be sent as a resource-id.</p>
INVALID_MESSAGE	Invalid message	The request could not be parsed.
INVALID_PATTERN	Invalid Pattern	<p>A pattern could not be parsed. The message should be sent upon first encountering the error.</p> <p>The ID of the layer pattern should also be provided as a resource-id.</p>
PATTERN_NOT_FOUND	Pattern not found	<p>The requested pattern could not be found.</p> <p>The id of the pattern that was requested should be sent as a resource-id.</p>
SET_JOB_RESOURCE_NOT_FOUND	Could not find requested resource for setting palletizing job	<p>One or more provided IDs in a set palletizing job telegram could not be found or recognized.</p> <p>The property name for the resource that was not found should be sent as a resource-id. For example, if no layer pattern with the provided ID exists, the value "layer_pattern_id" is included. If there are multiple errors, any of the conflicting parameters can be used.</p>
SET_JOB_INVALID_PARAMETER	Invalid parameter for setting palletizing job	<p>One or more provided parameters in a set palletizing job telegram is invalid.</p> <p>The property name for the parameter that caused the error should be sent as a resource-id. For example, if start_index exceeds the number of containers, the value "start_index" is included. If there are multiple errors, any of the conflicting parameters can be used.</p>

## 7 Common Telegrams

See the standard equipment interface document [1] for common telegrams that a palletizing cell should implement.

### 7.1 Info-request Telegram (HLC <-> Cell)

**Telegram Type ID:** info-requests

**Description:** This overrides the info-request telegram found in the standard equipment interface [1] with an additional request type, namely the ability to request layer patterns from HLC or cell. The HLC or palletizing cell should send a corresponding telegram as a reply to the request.

These messages are primarily used for (re-)synchronization. They will not be used for e.g. requesting a state change for the cell.

Request type	Value	Sender	Response telegram
Layer Pattern	layer-patterns	HLC, Cell	<a href="#">Layer Patterns</a>

#### Properties

Property	Type	Description
request	String	Requested resource. Can be one of the values described above.

#### Example

```
{
  "header": {...},
  "request": "state"
}
```

### 7.2 Change Completed (Cell -> HLC)

**Telegram Type ID:** change-completed

**Description:** This telegram is transmitted by the palletizing cell to notify the HLC that a change is completed. It includes a change type and resource id that informs about the action that has been completed.

The change completed is sent with the specified combination of resource ID and one of the change types seen below.

Telegram that caused the change	Change type	Resource id
<a href="#">Add/Update Layer Pattern</a>	If a layer has been added: <i>layer-pattern-added</i>  If a layer has been updated: <i>layer-pattern-updated</i>	ID of the layer that has been added or updated.
<a href="#">Remove Layer Pattern</a>	<i>layer-pattern-removed</i>	ID of the layer that has been removed.
<a href="#">Set Palletizing Job</a>	<i>palletizing-job</i>	ID of the layer that the cell is ready to palletize.

## Properties

Property	Type	Description
type	String	The type of change completed (see available types above).
resource_id	String	Optional (see above of the change type requires an id).  The id of the resource associated with the change that has been completed.

## Example

```
{
  "header": {...},
  "type": "layer-pattern-updated",
  "resource_id": "1"
}
```

## 7.3 Container Updates and Management

### 7.3.1 Container Placed on Pallet (Cell -> HLC)

**Telegram Type ID:** container.palletized

**Description:** Telegram transmitted by the cell every time a container is placed on a pallet.

## Properties

Property	Type	Description
pallet_location_id	String	ID of the location of the pallet which the container was placed on.
layer_pattern_id	String	ID of the layer pattern being used.

item_id	String or null	An ID for the item tracked by the cell.
index	Integer	Index of which container coordinates were used for the given layer pattern.
height_offset	Integer	Height offset in mm from the top of the actual pallet (see <a href="#">The Pallet</a> ).
timestamp	Date	Timestamp of when the container was placed on the pallet.

### Example

```
{
  "header": {...},
  "pallet_location_id": "15",
  "layer_pattern_id": "1",
  "item_id": "15",
  "index": 5,
  "height_offset": 0,
  "timestamp": "2023-01-19T09:01:50Z"
}
```

## 8 Clean Layer Pattern Telegrams

### 8.1 Configuration

#### 8.1.1 Get Layer Patterns (Cell <-> HLC)

**Telegram Type ID:** layer-pattern.all

**Description:** Sent by either the cell or the HLC when requested through an [Info-request telegram](#).

If requested by the HLC, all layer patterns stored in the cell should be sent. The HLC can then check the patterns for issues and send any updates if needed through [Add/Update Layer pattern](#) and [Remove Layer Pattern](#).

If requested by the cell, the HLC will send all layer patterns that should be used by the cell. The cell can then modify its stored layer patterns as needed, e.g. by removing patterns not present in the list of telegrams sent by the HLC.

The cell should send a [Change Completed](#) telegram for each layer pattern that was modified as a result of processing the layer patterns. The type should be set according to the action (added/updated/removed).

A maximum of **5** layer patterns may be sent per message. If there are more than 5 layer patterns, they must be sent over multiple messages/fragments. The properties *num\_fragments* and *fragment* specify how many fragments will be sent and which "part" this fragment includes, respectively.

Example: There are a total of 7 patterns. The first fragment contains the first 5 patterns, and the second contains the last two. *num\_fragments* will have the value 2 for both telegrams, and *fragment* is incremented by 1 for each telegram.

*Note that each message must have a unique sequence number and must be acknowledged individually.*

See appendix [Layer Pattern](#) for more details regarding the layer layout.

#### Properties

Property	Type	Description
fragment	Integer	Specifies which fragment this message is, starting at 1. The messages are expected to be sent in the correct order (1, 2, 3, ...).
num_fragments	Integer	The number of fragments that will be sent.
patterns	List of <a href="#">LayerPattern</a>	List of layer patterns. See <a href="#">LayerPattern</a> .

#### Errors

The following error types can be returned for this request:

- INVALID\_PATTERN
- FRAGMENT\_MISSING

## Example

```
{
  "header": {...},
  "fragment": 1,
  "num_fragments": 1,
  "patterns": [
    {
      "id": "layer-1",
      "layer_name": "1 box",
      "pallet_type": "EURO",
      "use_slip_sheet": false,
      "container_length": 360,
      "container_width": 280,
      "container_height": 100,
      "positions": [
        {
          "index": 0,
          "x": 0,
          "y": 0,
          "r": 0.0
        }
      ]
    }
  ]
}
```

### 8.1.2 Add/Update Layer Pattern (HLC -> Cell)

**Telegram Type ID:** layer-pattern.update

**Description:** Telegram transmitted by the HLC. Sent when a layer pattern should be stored on the palletizing cell. If the ID of the pattern is already stored on the palletizing cell, it is replaced with the data from the telegram.

After adding/updating the telegram at the cell, a [Change Completed](#) telegram with the ID of the pattern is sent by the cell.

If an error occurred while updating the pattern (e.g. error parsing the message), an error message should be sent instead.

See appendix [Layer Pattern](#) for more details regarding the pattern layout.

#### Properties

Property	Type	Description
pattern	<a href="#">LayerPattern</a>	The configuration for the layer pattern. See <a href="#">LayerPattern</a> .

#### Errors

The following error types can be returned by the cell for this request:

- INVALID\_PATTERN

## Example

```
{
  "header": {...},
  "pattern": {
    "id": "1",
    "layer_name": "1 box",
    "pallet_type": "EURO",
    "use_slip_sheet": false,
    "container_length": 360,
    "container_width": 280,
    "container_height": 100,
    "positions": [
      {
        "index": 0,
        "x": 0,
        "y": 0,
        "r": 0.0
      }
    ]
  }
}
```

### 8.1.3 Remove Layer Pattern (HLC -> Cell)

**Telegram Type ID:** layer-pattern.remove

**Description:** Telegram transmitted by the HLC. Sent when a layer pattern should be removed on the palletizing cell.

After removing the telegram at the cell, a [Change Completed](#) telegram with the ID of the pattern is sent by the cell. If the layer pattern was not found on the cell, it should still send the completion telegram.

#### Properties

Property	Type	Description
id	String	ID of the layer pattern to remove.

#### Errors

The following error types can be returned by the cell for this request:

- PATTERN\_NOT\_FOUND

#### Example

```
{
  "header": {...},
  "id": "1"
}
```

## 8.2 Palletizing Actions and Updates

### 8.2.1 Set Palletizing Job (HLC -> Cell)

**Telegram Type ID:** palletize.layer

**Description:** Request from the HLC to set the layer pattern to be used for palletizing. The request is sent on a per-layer basis, where the palletizer will continue to palletize containers until *num\_containers* have been processed.

The palletizing cell must be in an Idle, Stopped or Completed state to accept the message. Otherwise, an error is returned.

After processing the instructions (without errors), a confirmation should be sent through the [Change Completed](#) telegram.

To start the palletizing job, the HLC will send a *Request state change* with state *Start*.

The palletizer keeps track of how many containers have been processed for the given request, but should send updates to the HLC through the [Container Placed on Pallet](#) telegram.

When the palletizing has palletized the *num\_containers* specified in the telegram, it should enter the Complete state and wait for a reset from the HLC (through the *Request State Change* telegram).

If a *Request State Change* telegram for states Stop or Abort is received, the palletizing cell should stop palletizing the current layer.

#### Properties

Property	Type	Description
layer_pattern_id	String	ID of the layer pattern to use.
height_offset	Integer	Height offset in mm from the top of the actual pallet. See also: <a href="#">The Pallet</a>
start_index	Integer	The container index to begin palletizing at. Used if the current layer already contains containers. Cannot be larger than the number of containers in the layer pattern.
num_containers	Integer	Number of containers to palletize. Used if only a subset of containers should be palletized. Cannot be larger than the number of remaining container slots.
input_buffer_id	String	ID of the input buffer to pick container(s) from. <b>Will be changed in future versions to support picking from multiple buffers.</b>
pallet_location_id	String	ID of the location of the pallet to place containers on.

## Errors

The following error types can be returned by the cell for this request:

- INVALID\_MESSAGE
- SET\_JOB\_RESOURCE\_NOT\_FOUND
- SET\_JOB\_INVALID\_PARAMETER
- INVALID\_STATE

## Example

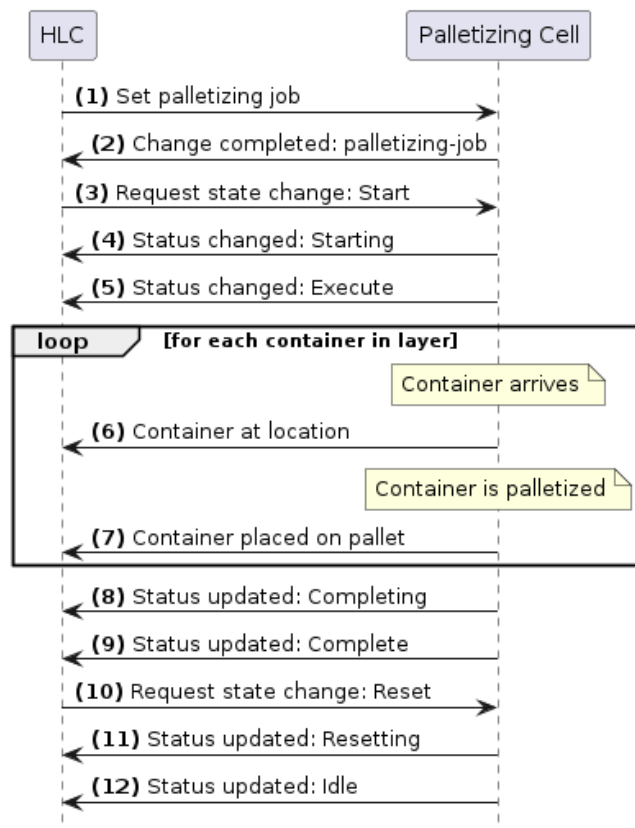
```
{  
  "header": {...},  
  "layer_pattern_id": "1",  
  "height_offset": 0,  
  "start_index": 0,  
  "num_containers": 7,  
  "input_buffer_id": "15",  
  "pallet_location_id": "15"  
}
```

## 9 Scenarios and Example Flows

This section provides sequence diagrams which define the order in which telegrams are sent between the HLC and palletizing cell in various scenarios. These sequences must be followed by the HLC and palletizing cell.

### 9.1 Palletizing a Layer

The HLC sends palletizing instructions to the palletizing cell per layer. Thus, the HLC handles supply flow of empty pallets and pickup of full pallets.



The following describes the flow shown above:

- (1) The HLC sends a [Set Palletizing Job](#) telegram to the palletizing cell. This telegram includes information for how many containers to palletize, which layout pattern to use and more.
- (2) The cell confirms that the job has been received and does not contain any errors (through the [Change Completed](#) telegram). This job will be used the next time the cell is in state Execute.
- (3) The HLC sends a *Request State Change* to the cell to start palletizing.
- (4) The cell enters the Starting state and prepares for palletizing.
- (5) The cell enters the Execute state and waits for containers to arrive. If a container is already present, it starts executing step (6).
- (6) A container arrives at the palletizing cell and is detected by a sensor. A *Container at Location* telegram is sent to the HLC.
- (7) After palletizing the container, the [Container Placed on Pallet](#) telegram is sent.
- (8) Once the cell has filled the layer with containers, it enters the Completing state.
- (9) The palletizing cell has completed the layer and is waiting to be reset.
- (10) The HLC sends a *Request State Change* to the cell to reset.

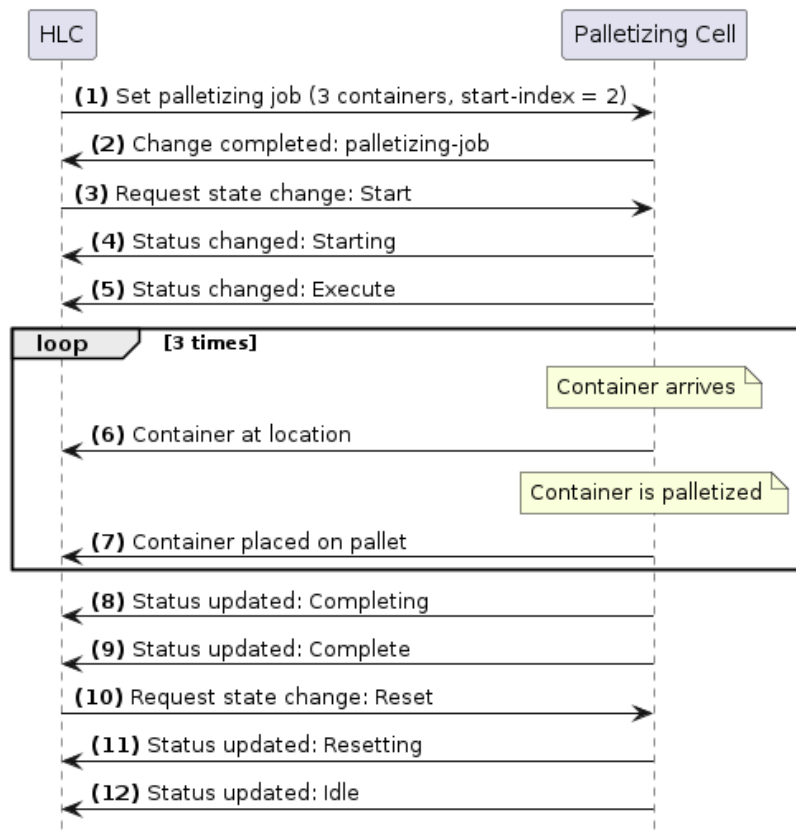
- (11) Cell starts resetting
- (12) Cell is idle and ready for new tasks.

The majority of the communication from the palletizing cell is to keep the HLC up to date with the current progress. The HLC can also use this information to estimate how often a container should be routed to the input buffer of the cell to maximize OEE.

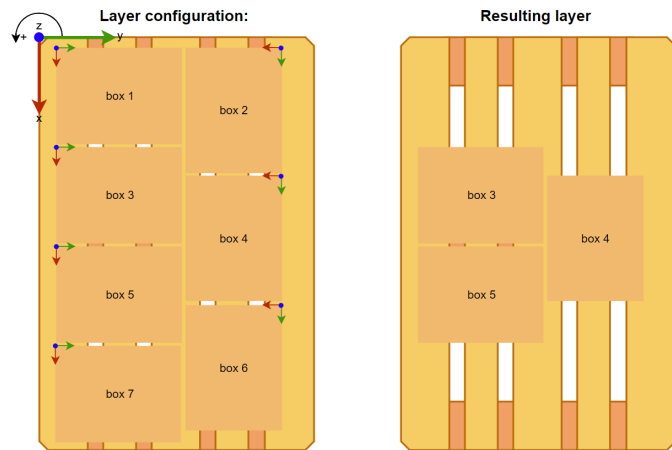
## 9.2 Palletizing a Partial Layer

Instead of palletizing an entire layer at a time, the HLC may instruct the palletizing cell to only palletize a subset of containers. This can be useful if only some of the containers are available. The remaining containers can thus be palletized at a later point, leaving the cell ready to work on other pallets.

Otherwise, palletizing a partial layer can be used for the top-layer.



The HLC sends a [Set Palletizing Job](#) telegram to the palletizing cell. This telegram includes information for how many containers to palletize, which layout pattern to use and from which index to start. In this example, three containers are to be palletized, starting from the index 2 (third container). For reference, see the image below. Afterwards, the job is confirmed by the cell, and the HLC requests the cell to start the job.



## 10 Appendix

### 10.1 Layer Pattern

This appendix provides an example of the concept of a layer pattern as well as how it can be used by the HLC.

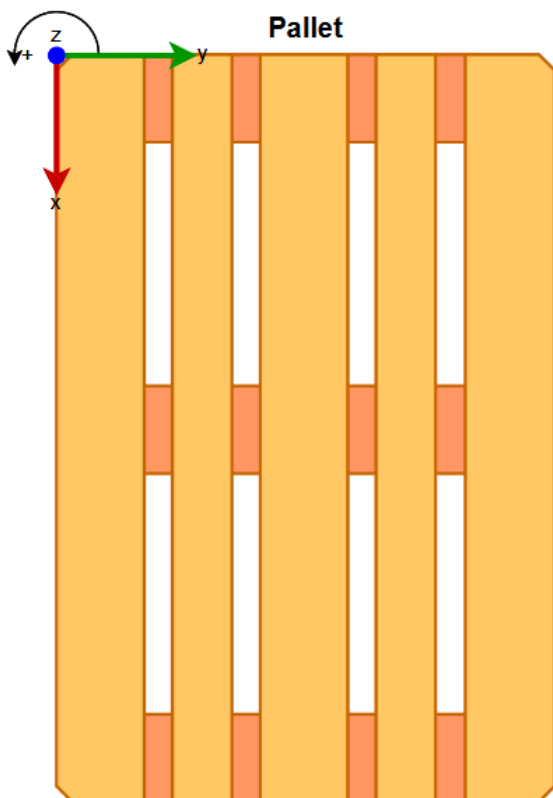
Coordinate systems are defined using a right-hand-coordinate system. Thus typically, the z-axis points upwards and has a positive rotation in the counterclockwise rotation.

All coordinates are provided in millimeters and rotations are in degrees.

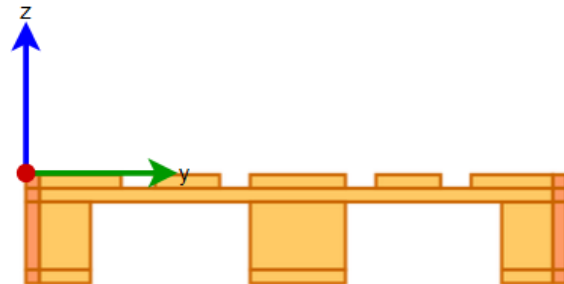
#### 10.1.1 The Pallet

A pallet has the coordinate system seen below. The origin is placed at the top-left corner of the pallet. Although the opposite corner (bottom-right seen in the image below) could likewise be used, it is important to be consistent throughout the palletization process.

Top view:



Side view:

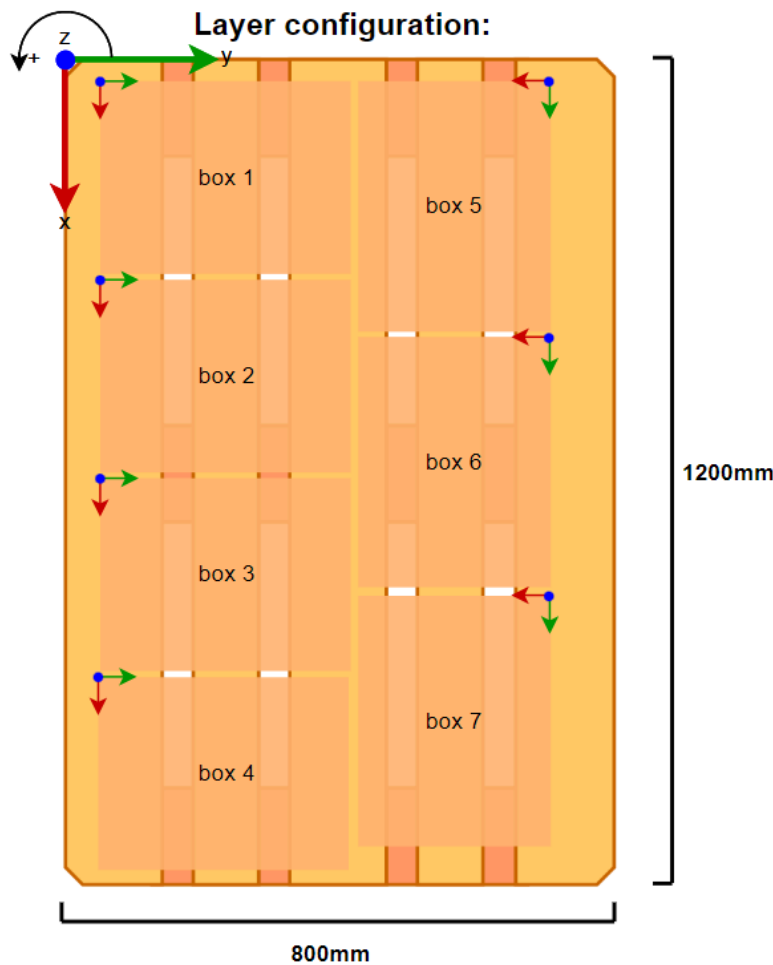


Pallets can be standard EURO pallets, US standard pallets, etc. Layers of stacked containers are positioned in the pallet's coordinate system.

### 10.1.2 Describing a Layer Pattern

Containers are palletized on pallets in a layer-by-layer manner. A layer has a specific configuration which defines the number of containers in the layer, and also the specific position and rotation of these containers in the layer's coordinate system.

The following is an example which illustrates a layer pattern with 7 containers.



As shown in the illustration, the layer pattern has a coordinate system in its top-left corner, same as the previously mentioned coordinate system for the pallet. In essence, the coordinates for the layer will be the same as for the pallet.

The boxes are placed in the layer's coordinate system. As an example, container 1 has the coordinates  $x = 30$  mm,  $y = 50$  mm, and rotation of 0 degrees, whereas container 7 has the coordinates  $x = 730$  mm,  $y = 645$  mm, and rotation of 270 degrees. Note that the point of reference for a container is also moved when a container is rotated, as shown in the figure.

A layer's pattern is stored in the following object:

```
{
  "header": {...},
  "layer_name": "7 normal boxes",
  "pallet_type": "EURO",
  "use_slip_sheet": false,
  "container_length": 360,
  "container_width": 280,
  "container_height": 100,
  "positions": [
    {
      "index": 0,
      "x": 30,
      "y": 50,
      "rotation": 0.0
    },
    ...
    {
      "index": 6,
      "x": 730,
      "y": 645,
      "rotation": 270.0
    }
  ]
}
```

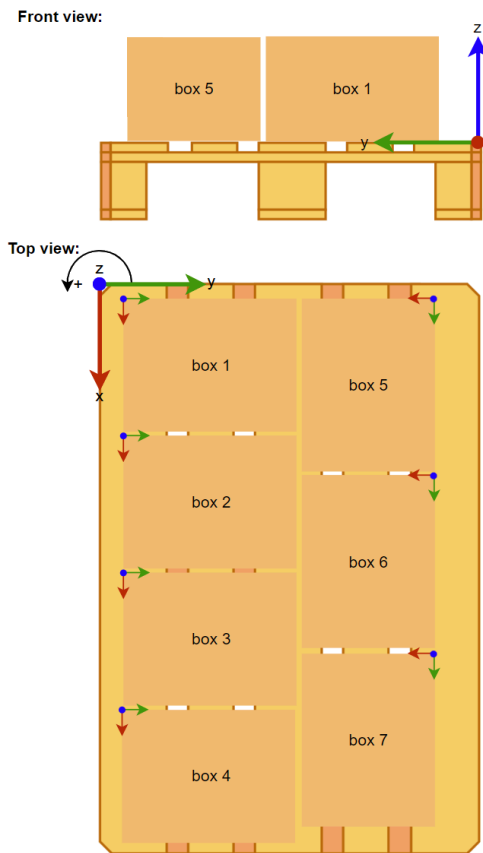
### 10.1.3 Using a Layer Pattern

The [Set Palletizing Job](#) telegram is sent when instructing the palletizing cell to palletize a single layer. In this telegram parameterize the layer pattern with the layer's position and orientation in the pallet's coordinate system.

If a euro pallet is used, then the following is an example of a [Set Palletizing Job](#) telegram.

```
{
  "header": {...},
  "layer_pattern_id": "1",
  "height_offset": 0, (the origin is located at the top of the pallet)
  "start_index": 0,
  "num_containers": 7,
  "input_buffer_id": "15",
  "pallet_location_id": "15"
}
```

The resulting palletized layer is the following:



Adding a second layer to the pallet the following telegram can be used:

```
{
  "header": {...},
  "layer_pattern_id": "1",
  "height_offset": 100, (layer 1 height)
  "start_index": 0,
  "num_containers": 7,
  "input_buffer_id": "15",
  "pallet_location_id": "15"
}
```

A third layer can be added to the pallet with only a subset of the layer's containers. The subset can be selected by setting the start-index and num-containers. The following telegram is an example of how this can be done.

```
{
  "header": {...},
  "layer_pattern_id": "1",
  "height_offset": 200, (layer 1 height + layer 2 height)
  "start_index": 2,
  "num_containers": 3,
  "input_buffer_id": "15",
  "pallet_location_id": "15"
}
```