Rescan Service Front-end Developer Documentation

Introduction

The RescanService is a service for handling rescans. Rescans are checks, performed by a store employee. They are used in a selfscanning environment to verify the shopping behaviour of a customer. Rescans can be successfull or failing. A successfull rescan means the customer scanned the items in his basket correctly. A failing rescan means the customer scanned something wrong. A rescan can be failing in two ways:

- The customer has items in his basket that were not scanned.
- The customer scanned items that are not in his basket.

Both can be detected by rescanning, but the last only by a full rescan.

There are two rescan types:

- **Partial** rescan: a quick check, the employee scans some items up to a specified minimum. When a difference is found before the minimum is reached, the typical next step is a full rescan.
- **Full** rescan: all items are scanned by the employee.

For more information, see the Introduction page.

For more information about the exact requests and responses and HTTP codes, see the Swagger documentation available when on the URI of the RescanService.

Calling the RescanService

Calls to the RescanService have the following URI format:

{BaseURI}/{APIversion}/{Controller}/{Command}

BaseURI: the protocol, server and port, for example: "http://localhost:7000" or "https://localhost:7000/rescanservice"

APIVersion: the version of the API, for example "v1" or "v2"

Controller: can be "common", "partial" or "full"

Command: see the Swagger documentation for available commands.

All request headers should have Content-Type : "application/json". All responses are in json format.

When errors occur, the response will contain error information as specified in RFC-7807.

Example scenarios

Minimal succesfull partial rescan scenario (with actual rescanning)	common/AvailableRescanForCustomerCard partial/Start partial/Add partial/Finish
Typical successfull partial rescan scenario	common/AvailableRescanForCustomerCard common/GetRescanOverview partial/Start partial/Add (repeated until MinimumRequirementsPassed = true) partial/Finish
Minimal failing partial rescan scenario (with FullContinue)	common/AvailableRescanForCustomerCard partial/Start partial/Add (returning DifferencesFound) partial/SwitchToFull full/Finish
Typical failing partial rescan scenario	common/AvailableRescanForCustomerCard common/GetRescanOverview partial/Start partial/Add (repeated X times) partial/SwitchToFull full/Add (repeated Y times) full/GetDifferences full/Finish
Minimal full rescan scenario	common/AvailableRescanForCustomerCard full/Start full/Add full/Finish
Typical full rescan scenario	common/AvailableRescanForCustomerCard common/GetRescanOverview full/Start full/Add (repeated N times) full/GetDifferences full/Finish

Full rescan with Reset	common/AvailableRescanForCustomerCard common/GetRescanOverview full/Start full/Add full/Reset full/Start full/Add full/Finish
Partial rescan with Reset	common/AvailableRescanForCustomerCard common/GetRescanOverview partial/Start partial/Add partial/Reset partial/Start partial/Add partial/Finish
Partial rescan with continued items	common/AvailableRescans common/GetRescanOverview partial/Start partial/Add (repeated X times until DifferenceFound = true) partial/ContinueOnError (same barcode as previous Add) partial/Finish
Partial rescan with Cancel	common/AvailableRescans common/GetRescanOverview partial/Start partial/Add partial/Cancel
Full rescan with Cancel	common/AvailableRescans common/GetRescanOverview full/Start full/Add full/Cancel
Full rescan with packaging item (also possible for partial rescan)	common/AvailableRescanForCustomerCard common/GetRescanOverview full/Start full/Add (returning PackagingRequired and AvailablePackagings) full/Add (same barcode but with selected packaging)

Getting RescanRequest for (scanned) customer card

A store's employee has the ability to get the rescan request for a (scanned) customer card by requesting common/AvailableRescanForCustomerCard.

API:

End point	Request Body
common/AvailableRescanForCustomerCard	• storeId : string • cardNumber: string • employeeId : string

The response of common/AvailableRescanForCustomerCard contains the rescanRequestId and the rescanType. The rescanType is required information to determine the URI of a Start call, such as partial/start or full/start. The rescanRequestId is required to identify the request for which to start a rescan session. The customerLanguage in the response can be useful information to inform the customer about the rescan.

Getting All Available Rescans

A store's employee has the ability to get all available rescan requests in a store by requesting common/AvailableRescans, that will show a list which can be

filtered using the *inProgress* flag and *storeId*.

- When "inProgress" is not set, then this call will show the requests that can be picked for a rescan.
- When "inProgress" is set, this call will show the requests which have active rescan sessions at this moment
- When "inProgress" is not specified, this call will return all rescanRequests. The response has an "inProgress" flag indicating a rescan session is active for the request, which can be checked.

When a rescan request's inProgress flag is set, any attempt to start a partial or full rescan will result in an error "RescanInProgress" .

API:

End point	Request Body
common/AvailableRescans	• storeId : string • inProgress: boolean • employeeId : string

The response of common/availableRescans contains the rescanRequestId and the rescanType. The rescanType is required information to determine the URI of a Start call, such as partial/start or full/start. The rescanRequestId is required to identify the request for which to start a rescan session. The customerLanguage in the response can be useful information to inform the customer about the rescan.

API Scenarios

Request Body	Response Body
{ "storeId": 50, "inProgress": true, "employeeId": "893489348", }	[{ "rescanRequestId": "3b6c4541-bbad-4cfe-b44f-0c218587d987", "storeId": "50", "basketId": "upBOCu1cZUG7cPOzrFlwJg", "cardNumber": "012345", "rescanType": "Full", "posNumber": 5, "inProgress": true, "customerLanguage": "EN", "timestamp": "20220306142245" }]

Getting the details of a rescan request before starting

API:

End point	Request body
common/GetRescanOverview	• rescanRequestId: string • employeeId: string

The AvailableRescans call returns a list with available rescans and a minimum of information for every rescan. If the employee wants to see more details of the rescan request, the common/GetRescanOverview can be used. This call returns all kind of details that can help the employee to be polite and to identify which items to scan first (in a partial rescan). The rescanRequestId is returned by AvailableRescans (so AvailableRescans has to called before GetRescanOverview).

GetRescanOverview can also be called after starting a rescan session, but it will return information based on the rescan request, not on the rescan session. For example, it will return the rescanType in the request. The actual rescan type in the session can be different when the employee switched to full rescan. Also the totals returned by GetRescanOverview are the totals of the original customer ticket, not the totals during a rescan session.

Starting partial or full rescan

The AvailableRescan returns the rescanType in the response. When the rescanType is "Partial", a partial rescan should be started. When the rescanType is "Full", a full rescan should be started.

API:

End point	Request body
partial/Start or full/Start	• rescanRequestId: string • language: string • employeeId: string

The rescanRequestId is the ID of the selected rescan in the available rescans.

The language is the preferred language of the employee.

The employeeId the card number of the employee.

A successful response contains the new sessionId and rescanState "Started".

The partial/start and full/start commands return a sessionId. This sessionId is an unique identifier for the rescan session. It has to be specified for every following command, such as Add, Update, Finish, etc.

After finishing the sessionId becomes empty. Note that in a session started for a partial rescan the commands for a full rescan are not allowed, and vice versa (the commands for a partial rescan are not allowed in a full rescan session).

RescanState

The rescan session returns a rescanState that can be one of:

- NotInitialized
- Starting
- Started
- Scanning
- Finishing
- Finished

The rescanState can be helpful to determine the correct flow.

- Adding items is only possible when the rescanState is "Started" or "Scanning".
- Removing or updating items is only possible when the rescanState is "Scanning".
- When the rescanState is "NotInitialized", the only possible actions are "Start", "Skip" or "GetRescanOverview".
- When the rescanState is "Finished", there are no more actions possible on the rescan request or rescan session.

Generic response properties

The following properties are returned in almost all responses from session operations:

- sessionId
- rescanRequestId
- basketView
- rescanType

- rescanState
- differencesFound
- finishAllowed

For partial rescans, the following properties are also returned:

- progressData
- minimumRequirementsPassed

BasketView

Most session operations return a basketView. This basketView represents the items scanned by the employee and is empty when starting the rescan. Depending on the configuration, the basketView can be compressed or uncompressed. In compressed basketViews equal items are presented on the same line. The basketView also contains several totals for item counts and various prices and discounts. A basketView contains items and items can have attributes. The following attributes have a special meaning in a front-end application:

- visible: if not set, the item should not be displayed
- ageRequired: if > 0, the customer age should be checked, this could be ignored during rescan
- updateQuantityAllowed: if not set, no updates are allowed on this item
- removeAllowed: if not set, the item cannot be removed
- askQuantity: should be ignored at this moment

Item messages and additional information are returned but could be ignore in a front-end rescan application.

The property "scannedByEmployeeNotByCustomer" gives an indication of the found differences in the basketView. In a front-end application, these items could be highlighted.

During a full rescan, the items in the basketView have the itemId's that can be used in an Update or Remove call.

AddedItem and addedItemView

A front-end application could also use the returned addedItem instead of the complete basketView, or use the addedItem in partial rescan and a basketView in full rescan.

AddedItem contains only the details of the item last added. AddedItemView is the basketView line with the added item. Depending on the compression type and the number of equal items already scanned,

the quantity in an addedItemView may be different from the quantity in addedItem.

AddedItem can be useful to create user interfaces with less details.

Adding items

API:

End point	Request body
partial/Add or full/Add	• sessionId: string • barcode: string • quantity: int • packaging: Packaging object

Packaging objects are returned by the RescanService when the the HTTP response is code 422 with a message "PackagingRequired" or "InvalidPackaging" in the response body. The end-user can select a packaging and the packaging object can be passed again to the Add call.

The packaging types are:

Number	Name	Description
0	None	No packaging
1	Container	The item is sold in a container, such as a crate. If the container is filled to its capacity, the back end might decide to convert the bottles and the container to a FullContainer.
2	FullContainer	The items are sold in a full container, meaning the container is filled up to its capacity.

3	Multipack	The item is part of a multipack, the customer may choose to buy a full pack or less items.

During a partial rescan the details of the last scanned item are returned in addedItem and addedItemView, such as the item description and the price.

Handling items with packagings

When an item requires packaging, the response of the Add (in partial or full rescan) will be HTTP error code 422 (UnprocessableEntity) with a response body. The response body contains a result message "PackagingRequired" and a list of available packagings.

For example, when the request is:

{

"sessionId": "Xd7iV70xmEGmqdm-Fk79eg",

"barcode": "41030806"

}

the response can be:

```
{
  "availablePackagings": [
    {
      "type": "None",
      "capacity": 999,
      "barcode": "1234",
      "descriptions": []
    },
    {
      "type": "Container",
      "capacity": 24,
      "barcode": "32",
      "descriptions": [
        {
          "description": "PFAND CONTAINER OF 24",
          "language": "DE",
          "type": "Short"
        }
      ]
    },
    {
      "type": "Container",
      "capacity": 6,
      "barcode": "41",
      "descriptions": [
        {
          "description": "PFAND CONTAINER OF 6",
          "language": "DE",
          "type": "Short"
        }
      ]
    },
    {
      "type": "Container",
      "capacity": 24,
      "barcode": "16",
      "descriptions": [
        {
          "description": "PFAND CONTAINER OF 24 Special Edition",
          "language": "DE",
          "type": "Short"
        }
      ]
    }
  ],
  "itemMessage": "",
  "type": "https://tools.ietf.org/html/rfc4918#section-11.2",
  "title": "Unprocessable Entity",
  "status": 422,
  "detail": "PackagingRequired"
}
Response headers
```


This response shows 3 available packagings: a "None" packaging, a container with 24 bottles and a container with 6 bottles.

To specify a container with 6 bottles, use:

{

"sessionId": "Xd7iV70xmEGmqdm-Fk79eg",

"barcode": "41030806",

"packaging" : { "type": 1, "capacity": 6, "barcode": "41", "descriptions": [{ "description": "PFAND CONTAINER OF 6", "language": "DE", "type": 1 }

}

or

{

```
    "sessionId": "Xd7iV70xmEGmqdm-Fk79eg",

    "barcode": "41030806",

    "quantity": null,

    "packaging" : { "type": 1, "capacity": 6, "barcode": "41", "descriptions": [ { "description": "PFAND CONTAINER OF 6", "language": "DE", "type": 1 }

}
```


Note the returned basket view will contain 6 bottles and 1 packaging.

To add the item without packaging, specify a quantity=1 and do not specify a packaging or set it to null, like this:


```
{

    "sessionId": "Xd7iV70xmEGmqdm-Fk79eg",

    "barcode": "41030806",

    "quantity": 1

}
```


Another way to do this is to use packaging type "0" ("None"). A packaging with type "0" is a placeholder for "no packaging", for example: { "type": 0, "capacity": 999, "barcode": "1234", "descriptions": [] }

It can also be used to specify a bottle without packaging, for example:


```
{
   "sessionId": "J19CbB4bFEOxUuLjvMXLZA",
   "barcode": "41030806",
   "quantity": 1,
   "packaging": { "type": 0, "capacity": 999, "barcode": "1234", "descriptions": [] }
}
```


A container for 6 bottles but filled with just 1 bottle is:


```
{
   "sessionId": "x1xFtp9Z00uM-1jZiYgnaQ",
   "barcode": "41030806",
   "quantity": 1,
   "packaging": { "type": 1, "capacity": 6, "barcode": "41", "descriptions": [ { "description": "PFAND CONTAINER OF 6", "language": "DE", "type": 1 } ] }
}
```


When the response is "InvalidPackaging", the supplied packaging is not correct. Supply a packaging from the given packagings and add again.

Handling items with age requirements

When an item has an age requirement, the response of the Add will have the minimum required age as a value in the "ageRequired" attribute. It can be used, for example, to show a popup message.

Handling items with item messages

When an item has an item message, the response of the Add will have the number of the item message in the itemMessage property of the response, It can be used, for example, to show a popup message.

Minimum Requirements for partial rescan

Before a partial rescan starts, the common/GetRescanOverview can be called to retrieve the minItemsToScan and minAmountToScan values. After the rescan has been started the progressData field of the partial/Start response returns the remainingItems and remaingAmount. After every following partial /Add command these fields are updated and returned again.

The minimumRequirementsPassed flag in the response of partial/Add becomes "true" when the employee passed the minimum number of items or the minimum amount. It is possible to given an indication to the employee at that point, to signal he can stop the rescan. But he can also continue rescanning. When differences are found after the minimum requirements, the minimumRequirementsPassed flag will stay "true", but finishAllowed will become "false".

The RescanService checks both quantity **and** amount. This means minimumRequirementsPassed can be "false" even when remainingItems is zero (and remainingAmount is not zero).

Partial rescans with differences

If differences are found during a partial rescan, the property differencesFound on a partial/Add response will become "true". This means an action is required: typically the employee will switch to a full rescan, but it is also possible to Reset, Cancel or ContinueOnError. It is not possible to Update or Remove existing items during a partial rescan. ContinueOnError offers partially the same functionality as Update/Remove, but ContinueOnError is limited. Too many corrections during a partial rescan could be an indication the employee can not be trusted.

Resetting a rescan

API:

End point	Request body
partial/Reset or full/Reset	• sessionId: string

Resetting a rescan means the current rescan session is removed, but the rescan request is not removed. After a reset, it is possible to select the request again and start a new rescan session.

Cancelling a rescan

API:

End point	Request body
partial/Cancel or full/Cancel	• sessionId: string • posNumber: int

Cancelling a rescan means an empty ticket will be sent to the POS, for example when the customer does not like the rescan, abandons the cart and leaves the store in a hurry.

The posNumber is the number of the POS where the rescan is performed. A value "0" means "no change" compared to the POS where the customer scanned the EOT barcode.

Skipping a rescan

Skipping a rescan means the original customer ticket will be sent to the POS without checks or corrections. It can be done during a rescan session or before starting a rescan session.

When the rescan session has been started:

API:

End point	Request body
partial/Skip or full/Skip	• sessionId: string • posNumber: int

When the rescan was not started, it is possible to skip the rescan request by specifying the rescanRequestId:

API:

End point	Request body
common/Skip	• rescanRequestId: string • posNumber: int • employeeId: string

Continuing errors in partial rescan

API:

End point	Request body
partial/ContinueOnError	• sessionId: string • barcode: string

Another way of handling errors during a partial rescan is to continue them. This means the barcode causing the difference is handled in a special way and will not longer cause a difference.

The RescanService parameter "AddItemOnContinue" determines what happens with the continued items. If this parameter is "true", the continued items will be added to the customer ticket and the customer has to pay them.

If this parameter is "false", these items will not be added to the customer ticket and the rescan can be finished as if it were a succesfull rescan.

The maximum number of continued items is limited to the value PartialRescanFailedItemsThreshold. When items are continued after the threshold, an error will be returned. The employee has to switch to full rescan or reset the rescan and start again.

Switching a partial rescan to full rescan

API:

End point	Request body
partial/SwitchToFull	• sessionId: string

It is possible to switch a partial rescan session to a full rescan session with the partial/SwitchToFull command. It is not possible to switch a full rescan back to a partial rescan. A switch to full rescan does not change the sessionId, only the rescan type. After switching, all calls to the "partial" controller are not longer allowed, such as partial/Add, partial/Finish.

The RescanService parameter FullRescanOptions specifies what to do with items already scanned when switching to full rescan. If the value is "FullContinue", the items are retained. If the value is "FullRestart", the full rescan starts with an empty basket.

Switching from partial to full rescan is allowed anytime during the partial rescan.

Finishing a partial rescan

API:

End point	Request body
partial/Finish	• sessionId: string • posNumber: int

The most important indication to see when a partial rescan can be completed is the finishAllowed property in the response of a Start or Add command.

If finishAllowed is set, it is allowed to complete the partial rescan.

If finishAllowed is set, this means the minimum requirements were passed **and** no differences were found, or (as a special case) no items were scanned at all.

If minimumRequirementsPassed is set and no differences were found, it is allowed to complete the partial rescan, but it is also allowed to scan more items before finishing.

The posNumber is the number of the POS where the rescan is performed. A value "0" means "no change" compared to the POS where the customer scanned the EOT barcode.

If the partial rescan session contains continued items, these are handled as follows:

- If "AddItemOnContinue" is set, the continued items are added to the customer ticket and the customer has to pay them.
- If "AddItemOnContinue" is not set, the continued items are removed from the customer ticket and the customer does not have to pay them.

Finishing a full rescan

API:

End point	Request body
full/Finish	• sessionId: string • posNumber: int

During a full rescan it is allowed to finish the rescan anytime, but it is the responsibility of the employee to scan all items in the customer basket before finishing.

Normally the employee will see the differences before he finishes the rescan. These differences are returned by full/GetDifferences. After finishing a full rescan, the items scanned by the employee will completely **replace** the customer ticket (except for unrescannable items). If the employee scans more items than the customer, the customer has to pay them; if the employee scans less items than the customer, the customer does not have to pay them.

Removing items in full rescan

API:

End point	Request body
full/Remove	• sessionId: string • viewItemId: string

When items have to be completely removed, full/Remove can be called. It takes a sessionId and a viewItemId. The viewItemId can be retrieved from the basketView in the various responses.

Getting Differences

API:

End point	Request body
full/GetDifferences	• sessionId: string

When a full rescan has been completed the employee usually wants to see the differences. The full/GetDifferences call returns two types of differences:

- ItemsScannedByCustomerNotByEmployee (the "forgotten" or "overscanned" items)
- ItemsScannedByEmployeeNotByCustomer (the "stolen" or "underscanned" items)

These are presented with price, description and totals. Currently the presentation is only an uncompressed view.

Updating an Item's Quantity in Full Rescan

During a full rescan an employee has the ability to update an item's quantity by requesting API full/Update. SessionId, quantity and viewItemId are required to update the item.

- sessionId: is used to retrieve a rescan data from database.
- viewItemId: required to load item from employee's basketView. If successfully retrieved, it will proceed with updating the item with the new quantity supplied. If not retrieved successfully then an *ItemNotFound* error will be the result.
- quantity: is is positive and should be greater than zero. For quantity zero the employee should use the Remove API .

Notes :

- Increasing quantity in some cases is limited if a packaging was supplied as the quantity must be within the packaging capacity.

End point	Request Body
full/Update	• sessionId: string • quantity: int • viewItemId: string

API Scenario (using postman)

Request Body	Response Body (postman's example)
{ "quantity": 5, "sessionId": "EX: f3mwBnLeXUSJlWtp3bOt7Q", "viewItemId": "EX: P00024689" }	{ "sessionId": "f3mwBnLeXUSJlWtp3bOt7Q", "rescanState": "Scanning", "rescanType": "Full", "rescanRequestId": "3b6c4541-bbad-4cfe-b44f-0c218587d987", "basketView": { "items": [{ "barcode": "3504182920011", "id": "P00024689", "quantity": 5, "price": { "grossPrice": 24.95, "netPrice": 24.95, "deposit": 0.0, "itemDiscount": 0.0, "liquidationDiscount": 0.0, "promotionDiscount": 0.0, "points": 0, "promotionPoints": 0 }, "shortDescription": "EN-NORMAL ITEM 1", "longDescription": "EN-NORMAL ITEM 1", "attributes": { "ageRequired": 0, "updateQuantityAllowed": true, "removeAllowed": true, "askQuantity": false, "scaledPriceCode": "", "countItem": true, "visible": true, "custom": { "CatalogueCategory": "NORMAL" } }, "messages": [], "additionalInfo": { "detailedItemDescription": "", "allergyInformation": "", "nutritionalInformation": "", "additionalInformation": "", "advertisements": [] }, "scannedByEmployeeNotByCustomer": true }], "price": { "grossPrice": 24.95, "netPrice": 24.95, "deposit": 0.0, "itemDiscount": 0.0, "liquidationDiscount": 0.0, "promotionDiscount": 0.0, "points": 0, "promotionPoints": 0 }, "totalItems": 1, "totalQuantity": 5 }, "differencesFound": true }

Getting active rescan sessions and acquiring an active rescan session

In some cases the employee device has a problem, preventing the session from continuing. In that case the rescan session can be continued on another device. It is possible to "grab" an active rescan session on another device with /partial/AcquireRescan or full/AcquireRescan. AcquireRescan can also be used to switch employees during a rescan session. The sessionId in AcquireRescan can be retrieved from the response of common /GetActiveRescanSessions.

API:

End point	Request body
common/GetActiveRescanSessions	• storeId: string • employeeId : string

GetActiveRescanSessions returns a list with session details, including the sessionId and rescanType. One of these sessions should be selected and the sessionId has to be passed to partial/AcquireRescan or full/AcquireRescan, based on the sessionId. For the selection of the correct session, the following guidelines can be used:

- Scan the employee card to retrieve the employeeId of the old session. In many cases this will be sufficient to identify the session.
- If this is not sufficient, for example, when the employee card is shared among the employees, scan the customer card to retrieve the customer Id.

API:

End point	Request body
partial/AcquireRescan or full/AcquireRescan	• sessionId : string • newEmployeeId : string

After a succesfull AcquireRescan, the old sessionId can not longer be used (a "dead session"). Any calls to the RescanService with the old sessionId will be rejected. AcquireRescan does not remove or change the scanned items in the session.

The response or AcquireRescan contains the new sessionId.

Starting a full rescan on a POS

Sometimes the full recan can not be performed on the employee device, for example when there is not enough space for all items in the basket. In such cases it is possible to start a full rescan on a normal POS. The partial/PerformFullOnPos can do this:

API:

End point	Request body
partial/PerformFullOnPos	• sessionId : string • posAddress : string • posNumber : string

Unlocking the return wall

This is not a function of the RescanService (it is part of the ScannerWallService), but is documented here since it is called on the same address as the RescanService.

API:

End point	Request body
store/UnlockReturnWall	• storeNumber:string • wallNumber:int