Hazcheck®
DETECT

Functional Description
For Industry Cargo Screening Solution

EXIS TECHNOLOGIES

Exis Technologies Limited
October 2019

CONTENTS

1. Overview

This document outlines the functionality of a Software as a Service solution for Cargo Screening for the Container Industry.

1.1. Service Assumptions

- Hazcheck Detect (HCD) delivered as a software-as-a-service solution, hosted and maintained by Exis Technologies.
- Container Line system calls the HCD web service, with results exposed to Container Line users only. Exis will support Container Line to create the calls with consultancy support.
- Calls to the HCD web service queue a screening to run, and then return an acknowledgement. Screening will be immediately processed in real time. Screening results will be determined by the keyword/ rules entered by Container Line, Industry group or those provided by Exis.
- Libraries provided by Exis, defined in Appendix.
- Container Line handles all existing internal interfaces.
- HCD cargo screening service is the method for integration of screening results into Container Line internal systems.
- Exis is not responsible for following up the results of the screening; this would be part of the container line's own internal processes.
- Exis to provide web user interface so that users can enter and maintain data search terms/ keyword/ rules, such Container Line data is not provided by Exis.
- Where the Industry group agrees common/standard rules/queries these can be made available to all users of HCD.
- Screening input and output to be stored for up to 1 month for diagnostic purposes.

1.2. Microservices Overview

The Hazcheck Detect solution consists of a front-end web service API and backend microservices to process shipment screenings. The front-end API will include endpoints for managing libraries of search queries, for managing users and authorisation, triggering a screen, retrieving results, and configuring service hooks to receive automatic notification of screening results.

A screening is triggered by an API call to the endpoint from a downstream service belonging to the customer. The object is added to an asynchronous queue, and a response is immediately returned, acknowledging receipt of the object to be screened. The acknowledgment includes a unique screening identifier, which can be used to correlate the screen request with the results.
The backend microservices consume objects from the queue in real-time, first preparing the object for processing, then running each of the rules against the object in parallel. The results are collated, and any configured service hooks are called.

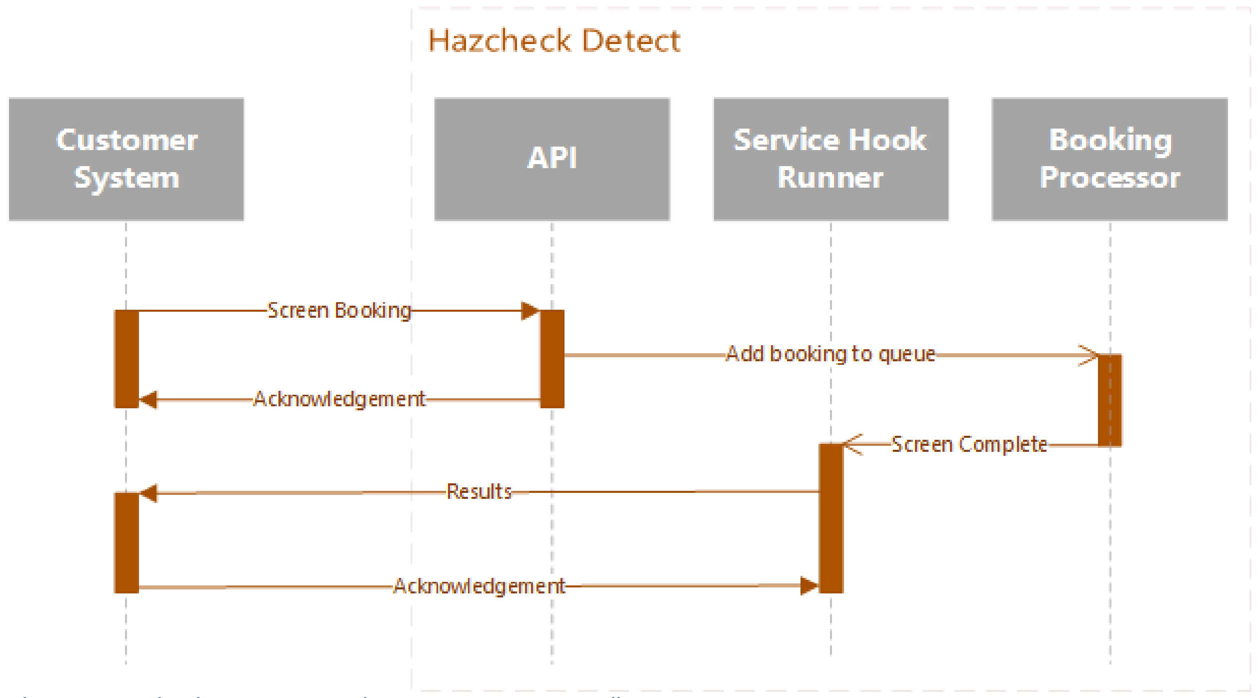


Figure 1:Hazcheck Detect screening process sequence diagram

2. Description

At its core, Hazcheck Detect is an engine for running custom rules against arbitrary objects and returning information about the rules that were triggered, and what caused them to trigger.

2.1. Security

All Hazcheck Detect endpoints are accessible via https only.

2.1.1. Authentication

Machine to machine connections are secured by oauth access tokens. User authentication will be via Single Sign On, with major identity federation protocols supported.

2.1.2. Authorisation

Hazcheck Detect associates a permission with every action. It will be possible to create custom roles with only the permissions you require and assign these roles to users. Additionally, each library will have its own set of permissions that apply only to itself and its rules. In this way you can allow users edit access to the rules of one library, read only access to another library, and no access at all to a third library.

2.2. Screening

2.2.1. Booking

Although Hazcheck Detect can screen any arbitrary object, it is useful to define a minimal structural schema for the booking object, in order that the rules provided by Exis can function. Additionally, any shared or industry wide rules would need to comply with the schema. See the Appendix for an example of the booking object schema.

2.2.2. Rules

In Hazcheck Detect, rules are the core component of functionality. A rule consists of a friendly name/description and the query, written in the Hazcheck query language. For more information on the Hazcheck query language, see the Appendix for the syntax guide.

2.2.3. Libraries

The organisational unit for collections of rules is the library. It will be possible to create libraries for whatever organisational reasons are required. The name of the library will be returned in the results whenever a rule from the library has been triggered.

2.2.4. Metadata

Both libraries and rules can contain metadata – that is, custom, flexible additional information that is not used or required by Hazcheck Detect itself but would be useful as part of the downstream processing. Some examples include priority, categorisation information, etc.
A hit result includes all metadata as a single collection, regardless of whether it came from the rule itself or the library containing the rule. One exception to this is when the library and the rule contain the same metadata entry – in these cases the rule value is used.

This means you can apply a piece of metadata once, on the library, and be confident that it will be returned when any of the contained rules are triggered. An example use case for this scenario is that you want all rules in a library to all be marked as “priority: Medium”, but you don’t want to have to set this value individually on each rule.
You can also override a library metadata value by applying the same piece of metadata to an individual rule. For example, in your library from the previous example, there may be a few rules that should be “priority: High”, instead of the default “priority: Medium”.

2.3. Exis Libraries

Exis will provide built-in rules that will be organised into the following libraries;

2.3.1. Undeclared Dangerous Goods

The undeclared DG rules would focus on items that are *not* declared as DG, trying to identify items that *should* have been declared as DG.

2.3.2. Mis-declared Dangerous Goods

The mis-declared DG rules would focus on items that *are* declared as DG, trying to identify items that have not been identified as the *correct* DG.

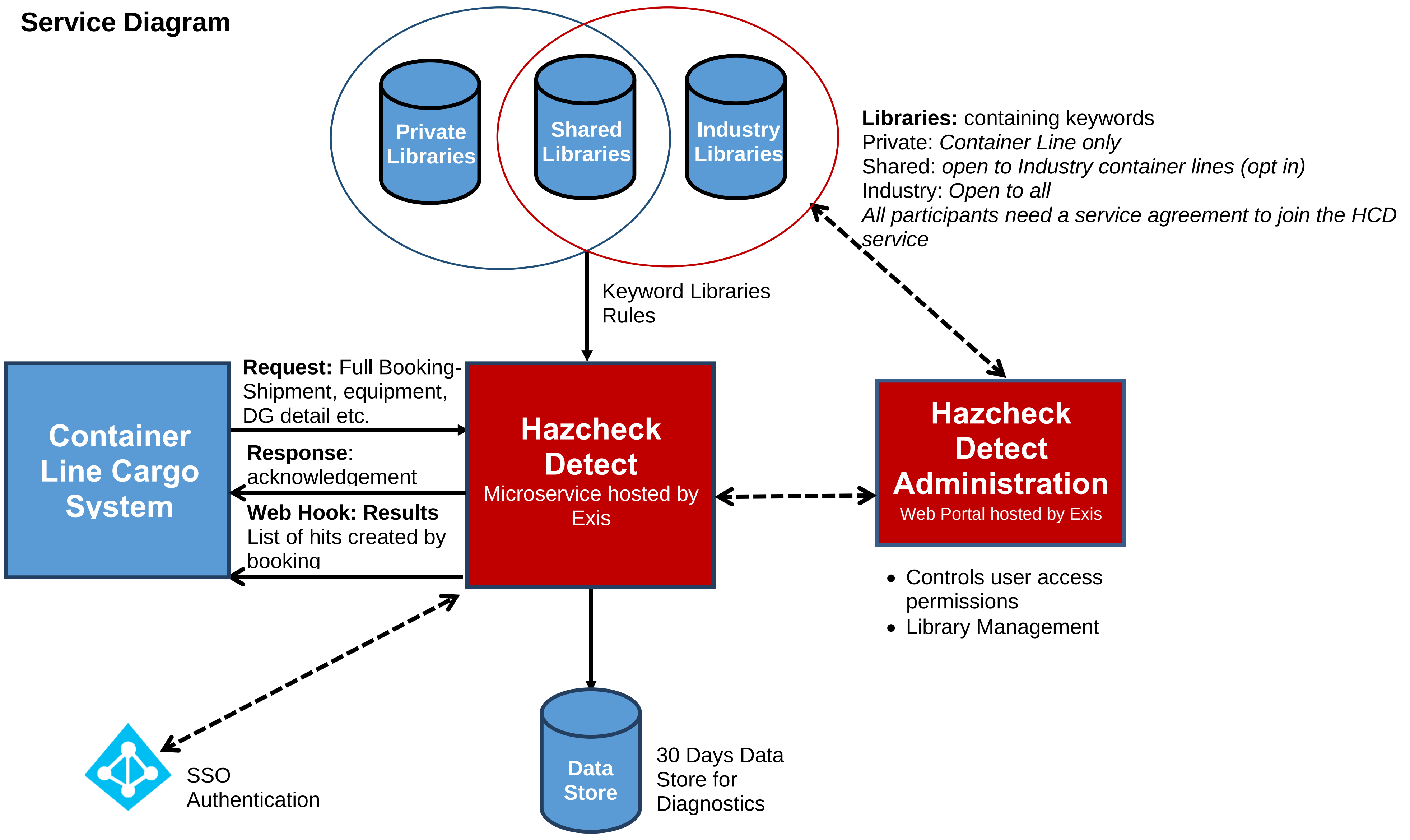
2.4. Sharing Rules

Hazcheck Detect will have a feature allowing Container Lines to share selected libraries of rules with other Container Lines using Hazcheck Detect (or their own screening system) and to consume libraries shared by those other Container Lines.

2.5. Service Hooks

Screen results will be available from an API endpoint once the cargo screen has completed, however it will also be possible to configure service hooks to notify when a cargo screen is complete. The service hook will also send the cargo screen results.

Service Diagram



Private Libraries
Shared Libraries
Industry Libraries
Libraries: containing keywords
Private: *Container Line only*
Shared: *open to Industry container lines (opt in)*
Industry: *Open to all*
All participants need a service agreement to join the HCD service
Keyword Libraries Rules
Container Line Cargo System
Request: Full Booking-Shipment, equipment, DG detail etc.
Response: acknowledgement
Web Hook: Results List of hits created by booking
Hazcheck Detect
Microservice hosted by Exis
Hazcheck Detect Administration
Web Portal hosted by Exis
• Controls user access permissions
• Library Management
SSO Authentication
Data Store
30 Days Data Store for Diagnostics