CRIF

TECH SPECS

Project Code: A0001

Project Name: Claims Portal - A2A interface – EL/PL Process

Version No.	3.1
Release Date	25/11/2022

Revision History

Release Date	Version	Description
08/05/2013	1.0	First description of the A2A interface issued
15/07/2013	1.1	Added two new exit codes in section 2.1.31 "Exit process"
02/08/2013	1.2	Clarification added to section 2.1.8 "Get hospitals list"
09/09/2013	1.3	- Clarification added to section 2.1.25 "Get Printable Document"
10/12/2013	1.4	- Clarification added to section 1.4.8.2 "Changes: about attachments" - Clarification added to section 2.1.31 "Exit Process": phase LiabilityAdmitted removed
17/12/2013	1.5	- Clarification added to section 2.1.21.1 "Add attachment"
22/01/2014	1.6	- Minor amendment to section 2.1.12.1: "Phase reached" updated
26/03/2014	1.7	- Word "flag" reworded as "Boolean flag" to better clarify the available options (true/false) - Clarification added to section 1.4.4.3 "Profiles and visibility of claims"
23/05/2014	1.8	- Clarification added to section 2.1.3.1 about the searchClaims() input - Clarification added to section 2.1.3.2 about the searchClaims() output
01/06/2016	1.9	Note added to section 2.1.3.2 about the searchClaims() output New fields and rejection process changes
05/07/2016	2.0	Amendments to section 2.1.9 and 2.1.10 for bulk transfer notifications
22/09/2016	2.1	Added the section 1.4 about the Instructions for developers
18/10/2016	2.2	Release 5 process versions Test case updated
13/03/2018	2.3	Release 6 Amendments for CR19 to section 2.1.23 for AddAttachments() max number of attachments.
11/05/2018	2.4	Release 6 Amendments for CR16 to sections: - 2.1.3.2 SearchClaim () added a field in the response - 2.1.5.2 GetClaimStatus() added a field in the response2.1.9.3 introduced the GDPR claim removed notification retrievable with thw GetnotificationsList()
15/06/2018	2.5	Release 6 Amendments for CR16 to sections: - 2.1.5.2 GetClaimStatus() Added the information about the new name of the field claimInfoResponse - 2.1.2.2 GetClaimsList() Added a field in the XSD response
08/10/2018	2.6	Release 6 amendments: - Amended reference to ELPL WSDL - Amended the process version - Added information about new "Advanced user" profile into Administrative console chapter (1.4.5.4) - Added a clarification for the getClaimStatus() response (2.1.5.2) - Added a clarification for the getNotificationList() input/output (2.1.9.2)
08/06/2020	2.7	Updated Instructions for developers section.
04/09/2020	2.8	Updated Instructions for developers section.
19/10/2022	2.9	08/08/2022: - Added paragraph Authentication (2.1) and modified all the call methods (from 2.2 on) to reflect the changes in the authentication process. - New Methods: GetToken (2.1.2), RefreshToken (2.1.3) and changePassword (2.1.4). - Updated the "Instruction for developers" section (page 11). - Updated paragraph Hints on the error handling (1.4.7). - Updated paragraphs 2.2.9 and 2.2.10 with the new specifications on how to retrieve and delete A2A notifications. 01/10/2022: - Updated minimum password criteria (2.1.4.2.1). - Updated ChangePassword() Specific notes (2.1.4.2.2). - Renamed input fields and namespace (2.1.1). - Updated paragraph Hints on the error handling (1.4.7).

		- Specific note on getNotificationList() and removeNotification() in case User = UserAsID (page 66).
25/10/2022	3.0	- Added clarifications of the Tokens behaviors and amended the example (from page 18 on). - Changed the validity periods in paragraphs from hours to seconds (2.1.2 and 2.1.3). - Changed the password expired behaviour (p. 22).
25/11/2022	3.1	- Added more clarifications of the Tokens behaviors and amended the provided example (from page 18 on). - Updated minimum password criteria (2.1.4.2.1).

Table of Contents

IMPORTANT INFORMATION

As a Developer you must ensure that you are aware of the clauses that your company has agreed to when registering as a user of the Claims Portal or a software house developing on behalf of Portal users, in particular clauses relating to the development of the User Interface and the requirement that your API is compliant with these technical specifications.

Instructions for developers

To ensure that important changes to the Claims Portal are not missed, sign up to the Claims Portal's Newsletter and receive regular communication updates.
https://www.claimsportal.org.uk/
Following is a list of points that must be reviewed prior to starting the development and adhered to during the development process. Other points are to assist the users with regard to the updating of certificates, the proper use of the Claims Portal via A2A systems and FAQ.

Calls frequency

The number of operations and calls made by your interface to the portal will depend on the number of claims managed by your organisation, but, when the instructions below are followed it is unnecessary to poll the portal more than around 220 times or less per hour. You must monitor the number of calls and if your application is polling more than 220 times per hour, you must establish the reason and make changes to bring the volume down to the required volume of hourly calls. CPL (Claims Portal Ltd) monitor the usage levels and the operations that are called and may contact users who are not compliant with these A2A specifications.
The portal is an interactive system it must not be used as if it where a batch system where calls are cached and then submitted to the portal in high volumes at one time. This practice impacts the performance of the service for all users. The objective must be to call the operations to manage the work flow as described below. User systems must not poll the portal with the same call multiple times every hour 24/7, the user interface must manage the workflow using the timelines indicated below.
All the operations that can be executed on the Claims Portal system can be distinguished in two main types:
1) Operations aimed to work on the claim (Addclaim(), AcceptClaim(), AddInterimSPFRequest(), etc…)
2) Operations aimed to obtain possible changes of the status of the claim (getClaim(), getClaimsList(), searchClaim(), etc…)
Operations that move the claims along the workflow (type 1) can be performed at any time.
Whilst all the operations needed only to update the information on the claims (type 2) must be spaced out. As the Claims Portal workflow is based on daily operations, it is not necessary to poll the system continuously to get updated information on the claim. The correct behaviour is to poll the system hourly.

Get the claim updated

When the user wants to know the current status of the claim, that is the phase in which the claim is in, the side where it sits and so on, they must use the searchClaim() functionality NOT the getClaim() call.

The searchClaim() call will obtain all the necessary information the user needs to know, to check the status of the claim and promptly respond to the other party. It is also mandatory that when the searchClaim() functionality is used by the user, it must be performed using always the corresponding search parameters (i.e. applicationID, sentDateFrom, sentDateTo, claimType…). This approach provides the best time responses from the system and obtains only the desired results for the user. You must not attempt to download all claims held in the portal for an organisation or all accompanying documentation, i.e. attachments, PDFs etc. If there is a need to download all claims due to the migration from web browser to A2A, please contact the Help Desk and ask for a copy of the Bulk Download Procedures.

Notifications

Notifications are generated by the Portal to inform the user of certain events, such as deadlines (timeouts) approaching, or where the other party has performed an action. Notifications indicating time remaining to make a decision on a claim at various phases in the process flow must be considered only a guide for the user in handling the claim. These notifications could change or may not be generated in the future and a users A2A system must not be developed that is dependent on these notifications.
Moreover, it is important that once the user reads the received notifications by the getNotificationsList(), they must remove them using the removeNotification() functionality.
It is the responsibility of the user to keep their own records of information submitted and received via the Portal. The user cannot rely on the Portal for any data storage purposes or use or rely on it as a CMS (claims management system).

Avoid hard-coding

In A2A Claims Portal the communication uses Web Service SOAP. The principal characteristic of a WS SOAP is to be completely independent from the technology\software behind, so you can write a WS Soap in any language you want running on any platform. During A2A development, it is mandatory that you follow the W3C Soap 1.1 specifications (https://www.w3.org/TR/2000/NOTE-SOAP-20000508/). You must not develop you're A2A system by hard-coding the SOAP xml output as it can slightly semantically change in the future (namespace prefixes, node order etc.): the A2A system must take into account logic content, not plain text representation.

Operations timeout

When an operation fails to be completed within the system timeout, the CMS must not be developed in a way that the system automatically re-tries to complete the call. The system must call the getClaimStatus() to know the current status of the claim and get the Process Engine Guid. In this way, the system knows whether the operation must be re-run or not and which Guid must be used for the next call.
The timeout for instances when no response is received from the portal should be set at 60 seconds.

Certificate renewal

Our security certificates are updated on an annual basis, it is important that the 2a systems automatically recognise and accept the new certificate without any interventions or modifications of the A2A system. However, users will be informed in advance when the new certificate will be installed and the certificate will be obtainable from the Help Desk if needed. To ensure that you are notified when a certificate will be deployed, sign up to the Claims Portal's Newsletter. https://www.claimsportal.org.uk/

Java version used

The Claims Portal system is based on java version 1.7. Using different version of Java should not impact the proper functioning of the system.

Insurer Indices

The Insurer Index for RTA claims is available on the Claims Portal web site https://www.claimsportal.org.uk/ in the developers section, for Claimant Representatives to cache if they so wish. If you use this feature you must ensure that you keep the cached index up to date by taking the latest version which is published on the web site on a regular basis.

Claims Portal do not issue the Insurer Index for EL and PL claims due to the frequency of the changes. Claimant Representatives must not create a cached version as it will become out of date very quickly, it may cause issues where claims are transferred and your cached version will not contain details of new users.

Probes

If you wish to use probes in the Live site to establish that the service is operational, ensure that you use the searchClaim() call and that you only poll the Live site a maximum of 62 times an hour. The searchClaims() must not retrieve any claim, it must be used only to check if the system is available or not.
Probes must not be used in the A2A Test site.

A2A Test/integration Site

The test site is available for users to test their development and changes to their API. It is not designed to handle high call volumes or performance testing. If you use automated testing software you must ensure that the volume is kept to the minimum required to validate that your application is working correctly and you must not submit the same call more times than is necessary to complete your tests. You must review the results and conduct further testing, if required. Once your testing is completed you must stop the automated testing tool and ensure that you are no longer connected to the site. Claims Portal monitor the usage of the site and may contact you if they believe that your testing is not compliant with these A2A specification and instructions.

A2A Credentials

To request A2A credentials for the test or production site you must be a registered user of the Claims Portal; a Claimant Representative, Compensator or a Software House/Third Party Developer. Complete the appropriate form available on the Claims Portal web site. https://www.claimsportal.org.uk/

Please Note: A2A credentials to the live site are not provided to Software Houses/Third Party's for their use or use by their client. Credentials for the live site are only sent to the authorised user of the Live site, the Claimant Representative or the Compensator.

As an authorised A2A user you must keep details of the credentials secure as you will need to provide them, excluding the password, if you make a support request.
You must not disclose them to anyone who is not authorised to use them on your behalf. If you believe that someone else has obtained possession of any of your identification details, you should immediately contact the Helpdesk. Security of the credentials is the responsibility of the organisation that is provided with the credentials.

Authentication/AsUser

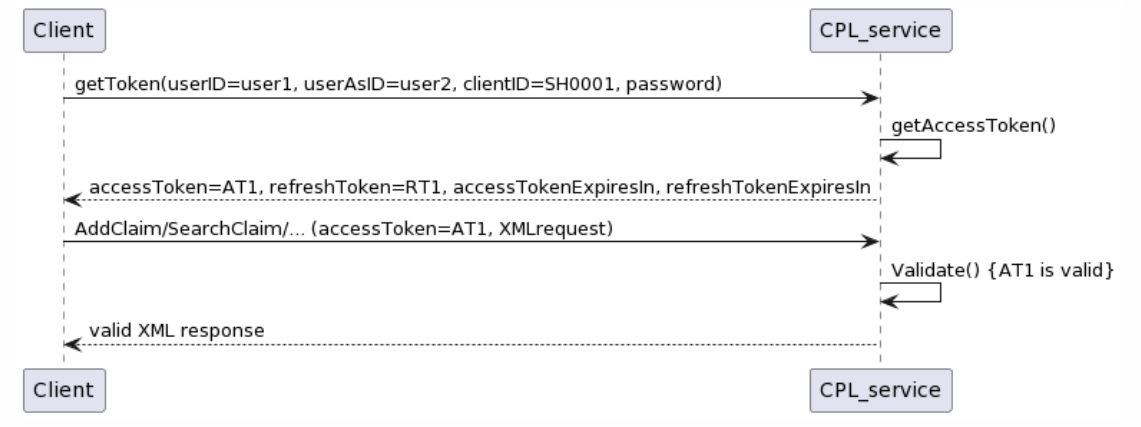
Please refer to section 2.1.1 Renamed input fields, for details of name changes.

We do not provide the AsUser for the Live site. It is the responsibility of the authorised user to create the AsUser. Please refer to "The Authentication" section which describes how the AsUser is constructed.

Tokens

getToken():

Before proceeding and making any of the calls available on the Portal a new accessToken must be retrieved from the system using the getToken() functionality, see paragraph 2.1.1. The accessToken retrieved by this functionality has a validity of 7200sec (2 hours) and must be included in every call to the Portal:



Together with the accessToken the getToken() functionality returns a field called refreshToken, this token must be used in the refreshToken() functionality to refresh the accessToken, the validity of this token is 86400sec (24 hours), (see paragraph **Errore. L'origine riferimento non è stata trovata.**)

The getToken() functionality returns also 2 fields called:

- accessTokenExpiresIn
- refreshTokenExpiresIn

These 2 fields contain the accessToken and refreshToken validity periods left until the expiration, expressed in seconds.

During the validity period of the accessToken (7200sec) the getToken() returns the same tokens, e.g.:

- getToken() with userID=user1, password=**** and UserAsID=user2 → accessToken1 (AT1), RefreshToken1 (RT1)
- getToken() within 2 hours with userID=user1, password=**** and UserAsID=user2 → AccessToken1 (AT1), RefreshToken1 (RT1)

Please note that the getToken() functionality should be used once every 86400sec (24 hours) to get the refreshToken and a new accessToken with a validity period set again to 86400sec (24 hours).
Calling the getToken() multiple times during the same day could cause confusion with the accessToken and the refreshToken validity periods increasing the possibility of login errors. For these reasons in order to refresh the tokens the refreshToken() functionality should be used.

refreshToken():

To refresh the accessToken AT1 the refreshToken() must be used. The refreshToken() RT1 can be used both before and after the expiration of the accessToken.

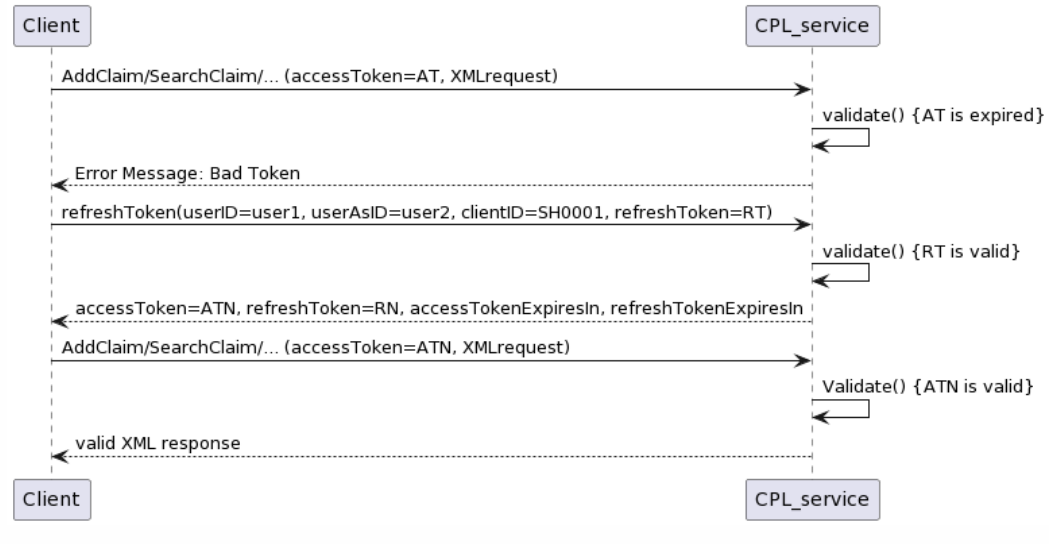
With the refreshToken(), together with a new accessToken with a validity 7200sec (2 hourse), a new refreshToken is also provided. This new refreshToken will have a validity of 86400sec (24 hours) minus the elapsed time from the first getToken() of the day, at this point the refreshToken used to perform the refreshToken() call is no longer valid.

As per the getToken() also the refreshToken() functionality returns 2 fields called:

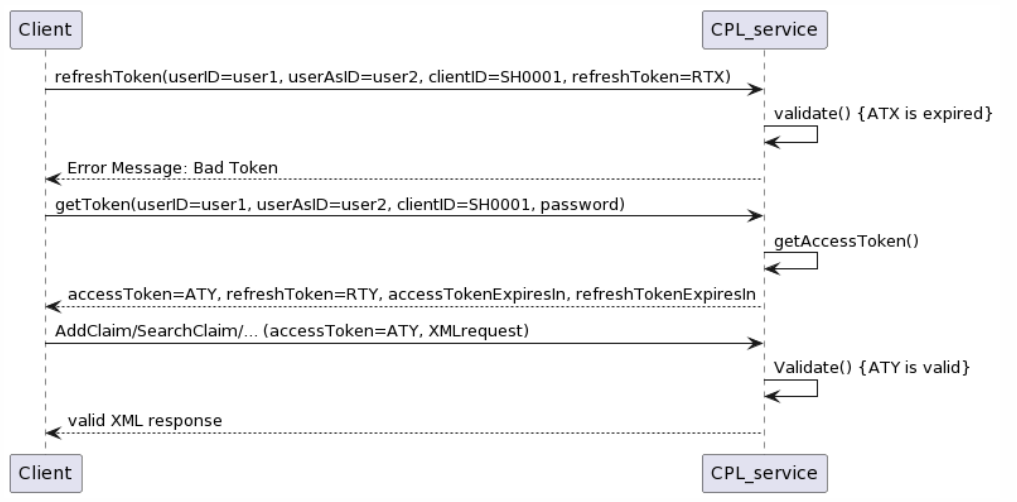
- accessTokenExpiresIn
- refreshTokenExpiresIn

These 2 fields contain the accessToken and refreshToken validity periods left until the expirations, expressed in seconds.
Please note that the validity of a new accessToken obtained with the refreshToken() functionality is always 7200sec (2 hours) whilst, as described above, the value of the resfreshTokenExpiresIn will always be less than 86400sec (24 hours) because it is the difference between 86400sec (24 hours) and the elapsed time from the last getToken() performed.

After 7200sec from the last performed getToken()/refreshToken() any call performed to the Claims Portal (addClaim(), getClaim(), searchClaim(),…) with the expired accessToken will return a “Bad Token”, to obtain a new accessToken a refreshToken() should be performed:

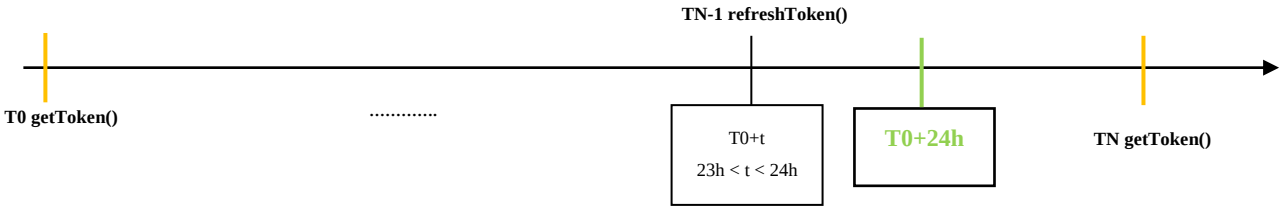


After 86400sec from the previous getToken() if a refreshToken() is performed the system wil return a "Bad Token", to obtain an accessToken valid again 86400sec a new getToken() is required:



Recommended Approach:

In order to avoid any token failure issue we recommend that you perform a refreshToken() immediately before the expiration of the refreshToken validity, in this way you will have enough time to perform another getToken() to reset to 86400sec (24 hours) the refreshToken validity as described below.



- Between T0+24h and TN, even if more than 86400sec (24 hours) have passed from T0, to call the Claims Portal system you can continue to use the accessToken obtained with the refreshToken() performed at TN-1, this is because the accessToken is always valid 7200sec (2 hours) from the last refreshToken() used to obtain it. You can exploit this time to obtain a new refreshToken without incurring "Bad Token" errors caused by performing a refreshToken() using an already expired refreshToken.
- After T0+24h, to obtain a new accessToken it is no longer possible to use the refreshToken obtained with the refreshToken() performed at TN-1, as more than 86400sec (24 hours) have passed from T0, the refreshTokenExpiresIn value is now 0 or lower than 0 and the system would return a "Bad Token" error if a refreshToken() is performed with an expired refreshToken.
- Between T0+24h and TN to obtain a new accessToken and to reset the validity of the refreshToken to 86400sec (24 hours) a new getToken() must be perfomed.
- Performing a getToken() between TN-1 and T0+24 (less than 86400sec (24 hours) from T0) would not reset the validity of the refreshToken as the refreshTokenExpiresIn would not have yet reached the 0.

A suggestion, in order to understand if a refreshToken() can be performed or if it is time to perform a new getToken() is to check the below conditions:

A. If accessTokenExpiresIn<= refreshTokenExpiresIn → refreshToken()
B. If accessTokenExpiresIn> refreshTokenExpiresIn → getToken()

Below is an example that explains this mechanism:

1. 13/07/2022 at 08:00:

   Perform a **getToken()** with userID=user1, password=**** and UserAsID=user2 → AccessToken1 (AT1), RefreshToken1 (RT1)

accessToken	accessToken validity	refreshToken	refreshToken validity
AT1	until 13/07/2022 10:00	RT1	until 14/07/2022 08:00

   accessTokenExpiresIn = 7200sec
   refreshTokenExpiresIn = 86400sec

2. 13/07/2022 at 09:45, 6300sec from pt.1:

   Check the following values:

accessTokenExpiresIn – time passed from pt.1 getToken() = 7200sec – 6300sec = 900sec
refreshTokenExpiresIn – time passed from pt.1 getToken() = 86400sec – 6300sec = 80100sec

Is condition A satisfied? YES
Is condition B satisfied? NO

Perform a **refreshToken()** with userID=user1, password=**** and UserAsID=user2 and refreshToken = RT1 → AccessToken2 (AT2), RefreshToken2 (RT2)

accessToken	accessToken validity	refreshToken	refreshToken validity
AT2	until 13/07/2022 11:45	RT2	until 14/07/2022 08:00
AT1	no longer valid	RT1	no longer valid

accessTokenExpiresIn = 7200sec
refreshTokenExpiresIn = 80100sec (86400sec – 6300sec)

3. 13/07/2022 at 11:30, 6300sec from pt.2 and 12600sec from pt.1:

   Check the following values:

   accessTokenExpiresIn – time passed from pt.2 refreshToken() = 7200sec – 6300sec = 900sec
   refreshTokenExpiresIn – time passed from pt.1 getToken() = 86400sec – 12600sec = 73800sec

   Is condition A satisfied? YES
   Is condition B satisfied? NO

   Perform a **refreshToken()** with userID=user1, password=**** and UserAsID=user2 and refreshToken = RT2 → AccessToken3 (AT3), RefreshToken3 (RT3)

accessToken	accessToken validity	refreshToken	refreshToken validity
AT3	until 13/07/2022 13:30	RT3	until 14/07/2022 08:00
AT2	no longer valid	RT2	no longer valid
AT1	no longer valid	RT1	no longer valid

   accessTokenExpiresIn = 7200sec
   refreshTokenExpiresIn = 73800sec (86400sec – 12600sec)

4. …
5. …

   During this period other refreshToken() calls can be performed, in this example we suppose that the refreshToken() at point 5 is performed 6300sec prior point 6 and 79200sec point 1.

6. 14/07/2022 at 07:45, 6300sec from pt.5 and 85500sec from pt.1:

   Check the following values:

   accessTokenExpiresIn – time passed from pt.5 refreshToken() = 7200sec – 6300sec = 900sec
   refreshTokenExpiresIn – time passed from pt.1 getToken() = 86400sec – 85500sec = 900sec

Is condition A satisfied? YES
Is condition B satisfied? NO

Perform a **refreshToken()** with userID=user1, password=**** and UserAsID=user2 and refreshToken = RT(N-2) → AccessTokenN-1 (ATN-1), RefreshTokenN-1(RTN-1)

accessToken	accessToken validity	refreshToken	refreshToken validity
AT(N-1)	until 14/07/2022 09:45	RT(N-1)	until 14/07/2022 08:00
…	…	…	…
AT3	no longer valid	RT3	no longer valid
AT2	no longer valid	RT2	no longer valid
AT1	no longer valid	RT1	no longer valid

accessTokenExpiresIn = 7200sec
refreshTokenExpiresIn = 900sec (86400sec – 85500sec)

At this point since refreshTokenExpiresIn<accessTokenExpiresIn, after 900sec from the refreshToken() a new getToken() must be performed.

7. 14/07/2022 at 09:30, 6300sec from pt.6 and 92700sec from pt.1:

   From point 6: refreshTokenExpiresIn<accessTokenExpiresIn

   Is condition A satisfied? NO
   Is condition B satisfied? YES

   More than 900sec have passed from point 6? YES.

   Perform a **getToken()** with userID=user1, password=**** and UserAsID=user2 → AccessTokenN (ATN), RefreshTokenN (RTN)

accessToken	accessToken validity	refreshToken	refreshToken validity
AT(N)	until 14/07/2022 11:30	RT(N)	until 15/07/2022 09:30
AT(N-1)	no longer valid	RT(N-1)	no longer valid
…	…	…	…
AT3	no longer valid	RT3	no longer valid
AT2	no longer valid	RT2	no longer valid
AT1	no longer valid	RT1	no longer valid

The accessToken and refreshToken are strictly linked to the userID and UserAsID used to obtain them, if you use more than one UserAsID you need to perform different getToken() and refreshToken() calls for each UserAsID.
If you need to use different userAsIDs to handle different tokens please ensure that you assign to the users the same profiles in order to always be able when using an accessToken generated with one userAsID, to retrieve all the available notifications even if generated on a claim handled by another userAsID.
Please refer to chapters **Errore. L'origine riferimento non è stata trovata.** and

Errore. L'origine riferimento non è stata trovata. in in order to understand how to handle the GetNotificationsList() and RemoveNotification() methods.

What to do in the case you "lose" the most recent accessToken() and refreshToken():

Our recommended approach if you have 'lost' the most recent accessToken and refreshToken returned by the system is to perform again a getToken(). As per specs:

During the validity period of the accessToken (7200sec) the getToken() returns the same tokens, e.g.:

- getToken() with userID=user1, password=**** and UserAsID=user2 → accessToken1 (AT1), RefreshToken1 (RT1)
- getToken() within 2 hours with userID=user1, password=**** and UserAsID=user2 → AccessToken1 (AT1), RefreshToken1 (RT1)

And this is valid regardless if the "lost" accessToken and refreshToken have been obtained with a getToken() or if the "lost" accessToken and refreshToken have been obtained with a refreshToken(). Please note that the only thing to bear in mind is that if the new getToken() is performed when the current accessToken is still valid the getToken() will not reset the tokens' validities:

Example 1:

- T0: getToken() → accessToken1, refreshToken1, accessTokenExpiresin=7200sec, refreshTokenExpiresIn=86400sec
  At some point within the 7200s of accessToken1 validity the tokens got lost
- T1 after 750s from T0: getToken() → accessToken1, refreshToken1, accessTokenExpiresin=6450sec, refreshTokenExpiresIn=85650sec

Example 2:

- T0: getToken() → accessToken1, refreshToken1, accessTokenExpiresin=7200sec, refreshTokenExpiresIn=86400sec
- T1 after 5400s T0: refreshToken() → accessToken2, refreshToken2, accessTokenExpiresin=7200sec, refreshTokenExpiresIn=81000sec
  At some point within the 7200s of accessToken2 validity the tokens got lost
- T2 after 750s from T1: getToken() → accessToken2, refreshToken2, accessTokenExpiresin=6450sec, refreshTokenExpiresIn=80250sec

Instead, if the accessToken is already expired, i.e. more than 7200sec have been passed from the last getToken()/refreshToken() performing a new getToken() will return a new couple of accessToken and refreshToken, with their validities reset to 7200sec and 86400sec:

Example 3:

- T0: getToken() → accessToken1, refreshToken1, accessTokenExpiresin=7200sec, refreshTokenExpiresIn=86400sec
  At some point within the 7200s of accessToken1 validity the tokens got lost
- T1 after 7300s from T0: getToken() → accessToken2, refreshToken2, accessTokenExpiresin=7200sec, refreshTokenExpiresIn=86400sec

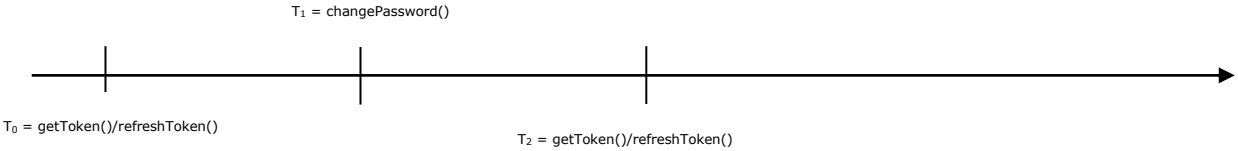
Example 4:

- T0: getToken() → accessToken1, refreshToken1, accessTokenExpiresin=7200sec, refreshTokenExpiresIn=86400sec
- T1 after 5400s T0: refreshToken() → accessToken2, refreshToken2, accessTokenExpiresin=7200sec, refreshTokenExpiresIn=81000sec
  At some point within the 7200s of accessToken2 validity the tokens got lost
- T2 after 7300s from T1: getToken() → accessToken3, refreshToken3, accessTokenExpiresin=7200sec, refreshTokenExpiresIn=86400sec

Please note that the above scenario should only been used in case of "lost" tokens, in a standard scenario we recommend to perform the getToken() once a day and use the refreshToken() to refresh the accessToken. Calling the getToken() multiple times during the same day could cause confusion with the accessToken and the refreshToken validation and increase the possibility of login errors.

Change password

The A2A password will expire every 90 days and a new A2A functionality to allow the users to change the A2A user's password has been introduced.
In order to reduce the risk of login errors the system allows the A2A to change password at any time regardless of the tokens validity. If at the time of the change password there are active tokens these will remain valid until their expiration datetime.
After the change password any new getToken()/refreshToken() must be performed using the new password.



T_0 = new token is obtained with a getToken()/refreshToken()
T_1 = password of the A2A user is changed using the changePassword() functionality
T_2 = new token is obtained with a getToken()/refreshToken() using the new password set with the changePassword() at T1()

Between T_1 and T_2 the calls can be performed with the token obtained with the getToken()/refreshToken() performed at T_0 (if the token has not yet expired).

You can exploit the remaining duration of the token to update the password in your application.

Change password doesn't require a valid token, **IT IS** possible to change the password when the token has already expired. **IT IS** also possible to change the password after the current password has expired. Refer to section 2.1.4ChangePassword for details.

Technical Support

The level of technical support available to developers may be found on the Claims Portal web site, in the Developers Section. Please ensure that you review the information available before you seek support. If you cannot find the answer to your query in the documentation provided, please complete the Contact us - A2A users / Software houses form available on the web site and email as an attachment to the Help Desk. Please ensure that you provide as much information as possible, including details of the credentials you are using **NOT the password**. This will enable support to investigate your query with minimal delay.

Identify New Attachments

Attachments are visible to the third party when the claim moves from one party to the other. However, if you need to identify whether any new attachment has been added by the third party whilst the claim remains with them, you should follow the process described below.

1. When you perform a searchClaim call you need to store for each claim the number of attachments at that point in time.
2. Once you have an attachment number assigned to a claim and when the number has changed you need to perform the GetAttachmentsList() for that claim, in this way you will be able to download, for each attachment, the following information:

- dataAttachmentGuid – the Id of the attachment
- dataAttachmentFileName – the filename of the attachment
- dataAttachmentTitle - a title for the file
- dataAttachmentDesc – a description of the file
- dataAttachmentOwner – the owner of the attachment
- dataAttachmentDate – the date of the attachment

3) For each new attachment you find in the GetAttachmentsListResponse you can perform the getAttachment() call using the dataAttachmentGuid to download the attachment

You should perform the above to ensure that you can identify any new attachment whenever you perform a searchClaim call. For those claims where you have not performed a daily searchClaim call you should make the searchClaim call to identify any new attachments. When the purpose of the call is to identify any new attachments, you must only make a searchClaim call once a day.

Introduction

1.1 Overview

This document describes the functionalities to access the Claims Portal system in A2A (Application-to-Application) mode.

1.2 References

- ***Glossary***

  [PIP] = PIP stands for Personal Injury Process. The product name is now:

  Claims Portal

  [CR] = Claimant Representative

  [CM] = Compensator

  [CMS] = Case Management System

  (the legacy in-house software interfacing with Claims Portal through A2A commands)

- ***References***

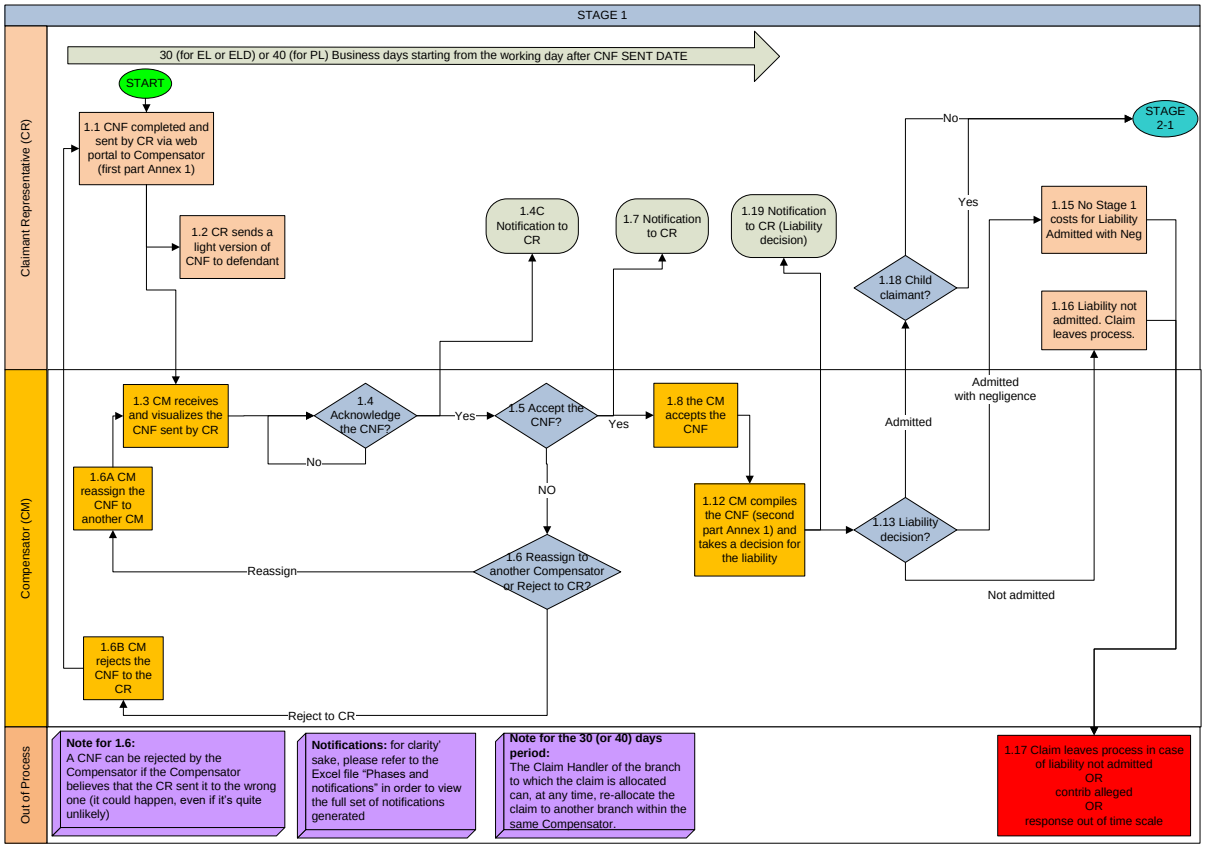
ELPLWS.wsdl– wsdl file

DocumentInput.xsd – schema used to add a Claim Notification Form
InsurerResponse.xsd – schema used to add an Insurer Response
GetClaimData.xsd - schema used to get the Claim details

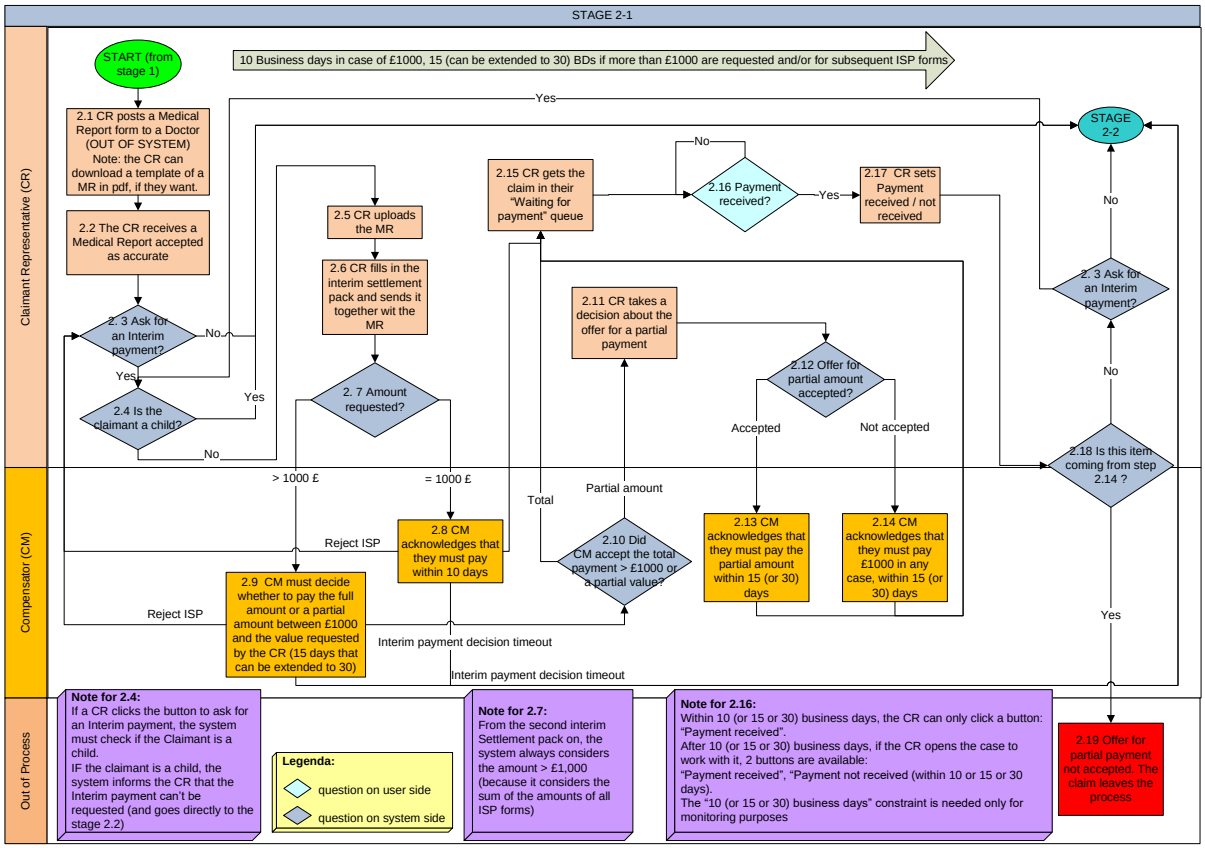
ELPL phases and notifications.xls – list of phases with descriptions related to the event that triggers a change of phase through the webUI of Claims Portal.
List of Notification messages that can be received by the system.

1.3 Workflows

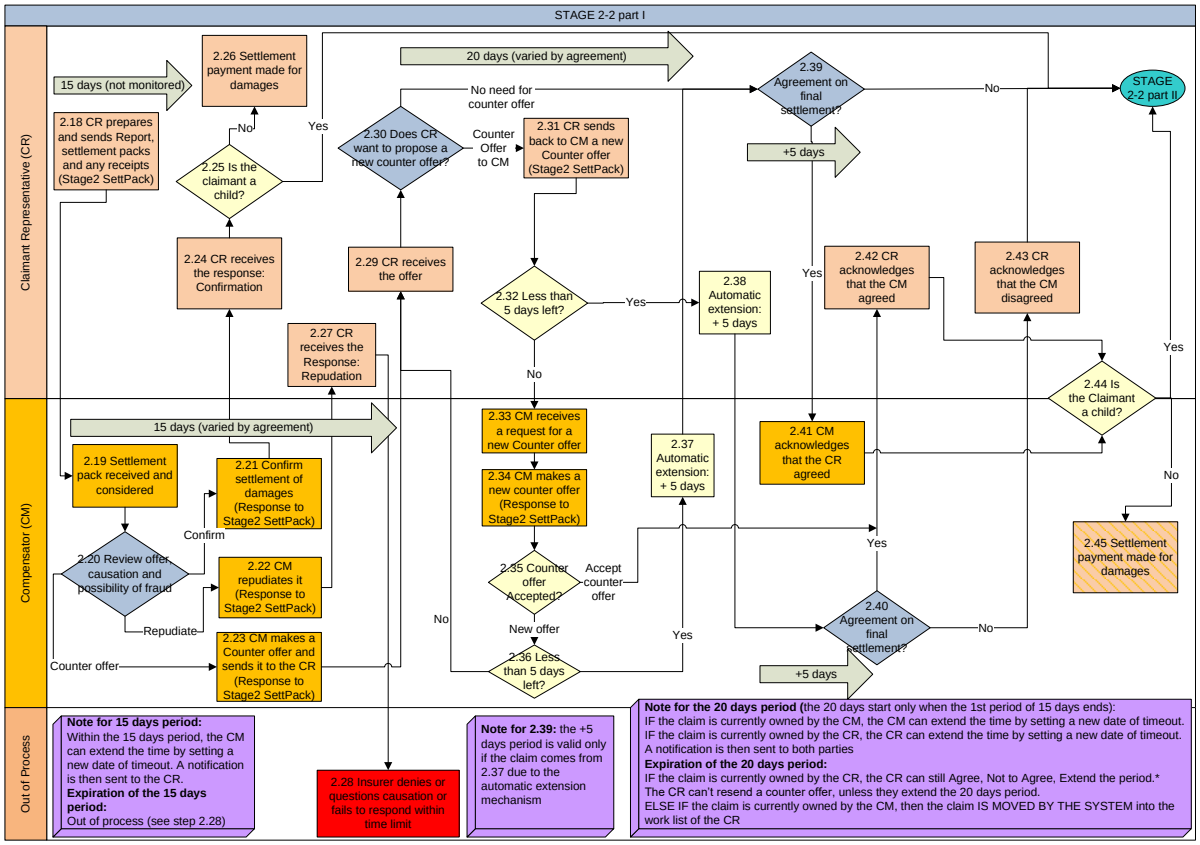
Stage 1



Stage 2.1


STAGE 2-1
Claimant Representative (CR)
Compensator (CW)
Out of Process
START (from stage 1)
10 Business days in case of £1000, 15 (can be extended to 30) BDs if more than £1000 are requested and/or for subsequent ISP forms
2.1 CR posts a Medical Report form to a Doctor (OUT OF SYSTEM) Note: the CR can download a template of a MR in pdf, if they want.
2.2 The CR receives a Medical Report accepted as accurate
2. 3 Ask for an Interim payment?
No
Yes
2.4 Is the claimant a child?
Yes
No
2.5 CR uploads the MR
2.6 CR fills in the interim settlement pack and sends it together wit the MR
2. 7 Amount requested?
> 1000 £
= 1000 £
Reject ISP
2.8 CM acknowledges that they must pay within 10 days
Reject ISP
2.9 CM must decide whether to pay the full amount or a partial amount between £1000 and the value requested by the CR (15 days that can be extended to 30)
Interim payment decision timeout
Interim payment decision timeout
2.15 CR gets the claim in their "Waiting for payment" queue
2.16 Payment received?
No
Yes
2.17 CR sets Payment received / not received
2.11 CR takes a decision about the offer for a partial payment
Total
Partial amount
2.10 Did CM accept the total payment > £1000 or a partial value?
2.12 Offer for partial amount accepted?
Accepted
Not accepted
2.13 CM acknowledges that they must pay the partial amount within 15 (or 30) days
2.14 CM acknowledges that they must pay £1000 in any case, within 15 (or 30) days
Yes
STAGE 2-2
No
2. 3 Ask for an Interim payment?
No
2.18 Is this item coming from step 2.14 ?
Yes
2.19 Offer for partial payment not accepted. The claim leaves the process
Note for 2.4:
If a CR clicks the button to ask for an Interim payment, the system must check if the Claimant is a child.
IF the claimant is a child, the system informs the CR that the Interim payment can't be requested (and goes directly to the stage 2.2)
Legenda:
question on user side
question on system side
Note for 2.7:
From the second interim Settlement pack on, the system always considers the amount > £1,000 (because it considers the sum of the amounts of all ISP forms)
Note for 2.16:
Within 10 (or 15 or 30) business days, the CR can only click a button: "Payment received".
After 10 (or 15 or 30) business days, if the CR opens the case to work with it, 2 buttons are available:
"Payment received", "Payment not received (within 10 or 15 or 30 days).
The "10 (or 15 or 30) business days" constraint is needed only for monitoring purposes

Stage 2.2


STAGE 2-2 part I
Claimant Representative (CR)
Compensator (CM)
Out of Process
15 days (not monitored)
2.26 Settlement payment made for damages
20 days (varied by agreement)
2.18 CR prepares and sends Report, settlement packs and any receipts (Stage2 SettPack)
No
Yes
2.25 Is the claimant a child?
2.30 Does CR want to propose a new counter offer?
No need for counter offer
Counter Offer to CM
2.31 CR sends back to CM a new Counter offer (Stage2 SettPack)
2.39 Agreement on final settlement?
No
STAGE 2-2 part II
+5 days
2.24 CR receives the response: Confirmation
2.29 CR receives the offer
2.32 Less than 5 days left?
Yes
2.38 Automatic extension: + 5 days
Yes
2.42 CR acknowledges that the CM agreed
2.43 CR acknowledges that the CM disagreed
Yes
2.27 CR receives the Response: Repudation
No
2.44 Is the Claimant a child?
15 days (varied by agreement)
2.33 CM receives a request for a new Counter offer
2.37 Automatic extension: + 5 days
2.41 CM acknowledges that the CR agreed
2.19 Settlement pack received and considered
2.21 Confirm settlement of damages (Response to Stage2 SettPack)
Confirm
2.34 CM makes a new counter offer (Response to Stage2 SettPack)
No
2.45 Settlement payment made for damages
2.20 Review offer causation and possibility of fraud
2.22 CM repudiates it (Response to Stage2 SettPack)
2.35 Counter offer Accepted?
Accept counter offer
Yes
Repudiate
New offer
No
2.40 Agreement on final settlement?
No
Counter offer
2.23 CM makes a Counter offer and sends it to the CR (Response to Stage2 SettPack)
2.36 Less than 5 days left?
Yes
+5 days
Note for 15 days period: Within the 15 days period, the CM can extend the time by setting a new date of timeout. A notification is then sent to the CR.
Expiration of the 15 days period: Out of process (see step 2.28)
2.28 Insurer denies or questions causation or fails to respond within time limit
Note for 2.39: the +5 days period is valid only if the claim comes from 2.37 due to the automatic extension mechanism
Note for the 20 days period (the 20 days start only when the 1st period of 15 days ends):
IF the claim is currently owned by the CM, the CM can extend the time by setting a new date of timeout.
IF the claim is currently owned by the CR, the CR can extend the time by setting a new date of timeout.
A notification is then sent to both parties
Expiration of the 20 days period:
IF the claim is currently owned by the CR, the CR can still Agree, Not to Agree, Extend the period.*
The CR can't resend a counter offer, unless they extend the 20 days period.
ELSE IF the claim is currently owned by the CM, then the claim IS MOVED BY THE SYSTEM into the work list of the CR

1.4 Constraints

1.4.1 Technical Constraints

All the functionalities will be provided on top of the functionalities provided by the flow platform product.

1.4.2 Security Constraints

HTTPS connections will be used to manage the security of the communications via SOAP message to the Claims Portal Web Services.

1.4.3 Workflow versioning constraints

1.4.3.1 Workflow process version

The claim workflow is implemented in the system by a workflow process.
When a new workflow is available in the system, that means that a new workflow process is deployed; each workflow process is identified by a workflow process version, which in detail has a VersionMajor and VersionMinor numbering: for example the first deployed workflow process has version VersionMajor=1 and VersionMinor=0 (this means workflow process version 1.0).
When a new workflow process version is deployed, previously deployed workflow process versions are not replaced or removed from the system, but are still available in the system to complete currently live claims started with previous workflow process versions.

1.4.3.2 Impact of deploying a new workflow process version to fix defects

When a new workflow process version is deployed in the system to fix some defects, this does not impact the available commands and their signature (their parameters).

1.4.3.3 Impact of deploying a new workflow process version implementing Change Requests

When a new workflow process version is deployed in the system to implement Change Requests, this will very likely impact the available commands and possibly the signature of some commands (by adding additional parameters).
Claims created before the deploy (with previous workflow process version) will not accept new commands or new parameters.
New commands will only be accepted by claims created after the deploy, using the new workflow process version.

1.4.3.4 Workflow process version used when creating a claim

When a new claim is created, the workflow process version used is always the latest available in the system, and this workflow process version will be used to manage the claim for the whole claim life, regardless of subsequently deployed workflow process versions.
Once a new workflow process version is deployed in the system, it is no longer possible to create claims with the old workflow process version.

1.4.3.5 Workflow process version used to progress an already existing claim

A claim maintains the version under which it was created. In order to update a claim or move it through the workflow, it is necessary to use the set of commands and schemas compatible with the process version under which the claim was created,

1.4.3.6 Multiple workflow process version living together in the system

When a new workflow process version is deployed in the system, new claims are created using the latest workflow process version, but claims previously created still alive in the system, will be completed using the workflow process version under which they were created.

1.4.3.7 Retrieving workflow process version

The workflow process version can be retrieved when calling SearchClaims() or GetClaimsList() commands: and since a claim is managed when it changes ownership, and this is done using SearchClaims()/GetClaimsList() to detect ownership changes (see section 1.4.8.5), to manage the process version there is no additional overhead when a claim is moved into the client software pipeline.
In case the workflow process version is not yet managed by the client, or in case should be retrieved the workflow process version for claims currently owned, it's possible to retrieve the workflow process version info for a specific claim by executing SearchClaims(criteria=ApplicationID), or SearchClaims(criteria=PhaseID) for claims in a specific workflow phase; see section 2.2.3 for details.

1.4.3.8 Manage Change-Request discontinuity using the process version

When a new workflow process version is deployed implementing Change Requests, it's critical that the client calling the A2A Commands is able to behave correctly for the claims before the Change Requests and also for the claims created after the Change Request implementation.
This means that, based on the workflow process version, some commands may be applicable or not, or the signature of some commands may be different (new parameters to be supplied).
It is important that the client should also retrieve and store the workflow process version, and use this info to properly manage the user interface, to enable or disable user interface elements which maps to workflow process version functionalities.

1.4.3.9 The table of Compatibility level

To track the discontinuity of workflow process versions in case of Change Request deploy, a Compatibility level table is maintained (see 1.4.14). The compatibility level is reported for each command and for phases (see subsections of section 2).

1.4.3.10 Policy in managing impact of new functionalities on the WSDL

New functionalities may be added in the future to the system, affecting the WSDL.
The policy is to minimize impacts on the WSDL, and maintaining a single WSDL is a priority. When possible, signature of existing commands may be extended to implement extended functionalities, otherwise a brand new command will be added to implement the extended functionalities. In case existing commands will become obsolete, they will be maintained in the WSDL, but only applicable to relevant workflow process versions.
Any exception to this policy will be evaluated to balance pros and cons.

1.4.4 Functional Constraints

1.4.4.1 Authentication

The authentication is performed using a specific web service method call invocation, this returns AccessToken and RefreshToken; AccessToken must be used on every web service method call invocation relative to the several functionalities; RefreshToken can be used for refreshing the validity time of the access token.

Claims Portal A2A authentication model requires two accounts, because the A2A user has the rights to access the A2A interface but cannot access alone any claim data; the second account is the web portal user, his/her claim data scope depends on the web portal user profile, and is not allowed alone to access the A2A interface.
This mechanism allows a single A2A account to act with different claim scope based on the web portal user profile specified in the A2A command. The A2A account should be considered as a kind of tunnel for the web portal account.

1.4.4.2 Profiles and allowed portal sections

Please note that each profile provides access to only one specific section:

Portal section	*Allowed profiles for the section*
Claims Portal (claim creation and/or processing)	**RTA & EL/PL** COMP Claim Handler COMP Claim Handler Team Leader COMP Branch Claim Handler COMP Claim Dispatcher COMP Branch Claim Dispatcher CR Claim Handler CR Team Leader **EL/PL only** COMP EL/PL Claim Handler COMP EL/PL Claim Handler Team Leader COMP EL/PL Branch Claim Handler COMP EL/PL Claim Dispatcher COMP EL/PL Branch Claim Dispatcher CR EL/PL Claim Handler CR EL/PL Team Leader **RTA only** COMP RTA Claim Handler COMP RTA Claim Handler Team Leader COMP RTA Branch Claim Handler COMP RTA Claim Dispatcher COMP RTA Branch Claim Dispatcher

	CR RTA Claim Handler CR RTA Team Leader
Administration (user creation and/or modification)	COMP Administrator CR Administrator ADVANCED User
Reporting	Not used

This means that the user used to obtain the accessToken should be a user with a profile allowed to access the Settlement section. It is the responsibility of the Authorised User to decide on which of their web portal users they will use as their UserAsID.

1.4.4.3 Profiles and visibility of claims (scope)

Please note that each profile has a different claim visibility, where the claim scope basically affects commands like GetClaimsList(), SearchClaims(), GetClaim().

This table shows examples of different scopes for these 3 functionalities:

Profile	*Command*	*Claims on which has previously worked (either closed or in progress or in other branches)*	*Claims assigned to the belonging branch*	*Claims assigned to other branches*	*Claims locked by someone else in the same branch*	*Claims locked by someone else in another branch*	*Claims under the ownership of the other side (CR for COMP, COMP for CR)**
COMP CH	GetClaimsList()	NO	YES	YES	NO	NO	NO
COMP BCH	GetClaimsList ()	NO	YES	NO	NO	NO	NO
COMP CH	SearchClaims()	YES	YES	YES	YES	YES	YES**
COMP BCH	SearchClaims()	YES	YES	NO	YES	NO	YES**
CR CH	GetClaimsList()	YES	n.a.	n.a.	NO	NO	NO
CR CH	SearchClaims()	YES	n.a.	n.a.	YES	YES	YES

* that have not been worked by the user stored in the field <UserAsID>

** FROM RELEASE 5 ON: only if the claim is assigned to the belonging branch of the user stored in the field <UserAsID> OR allocated to someone else in the same branch of the user stored in the field <UserAsID>

1.4.4.4 Workflow steps atomicity

A2A access and Web portal access share the same workflow, so a claim can be processed with an alternation of A2A commands and web forms, allowing each organisation to freely decide its claim management approach.
Please note that there are steps in the workflow which are made by few tabbed forms on the web portal: this means that in order to complete such workflow step the user should fill-in the fields in few tabbed forms; under this circumstance, no A2A command can be executed if the claim is stuck in the middle of two tabs.
Web forms and A2A commands behave as two paths which frequently cross each other: the next A2A command can be executed only when the web path reaches the intersection.

1.4.4.5 Concurrency consistency mechanism

In order to keep the system consistent, we have implemented a mechanism to make the clients always aware that there aren't concurrent actions on a claim.

Generally:
for each A2A action on a claim (even a simple GetClaim) the system sends back to the client 2 unique IDs:

- The applicationID of the claim
  [it does not change during the lifecycle of the claim]
- The "activityEngineGuid" representing the status in which **that** claim is currently
  [this is a unique ID representing "that claim in that specific moment". It's not banally an ID of a phase.
  CLARIFICATION: if 2 different claims are in the same phase of the workflow, they have 2 different activityEngineGUIDs

These IDs must be used by the A2A client to perform A2A actions on a claim.

The system will check if the Claim represented by the ClaimID is still in the phase represented by the activityEngineGuid:

- IF NOT, the action fails and the system sends back an error message
- IF YES, the action is successful and the system will give back to the A2A client a response containing (as stated above) also the claimID and the new "activityEngineGuid".

This mechanism is especially needed to avoid that an A2A client performs an action on a claim which is still in the same phase although it had been modified by someone else.

Example:

1) claim 123 is in phase A
2) the A2A client gets the claim 123 to read its details
3) someone else updates the claim which goes from phase A to phase B: A → B
4) someone else updates the claim which goes from phase B to phase A: B → A
5) the A2A client tries to update the claim to change its phase from A to C
   a. **without** this mechanism, the system would allow the A2A client to change the phase from A to C
   b. **with** this mechanism, the system informs the A2A client that "something happened to the claim after the last GetClaim performed". Hence the A2A client is forced to retrieve the claim (and the related "activityEngineGuid" to look at it before proceeding.

1.4.5 Description of the system design

The A2A system design is based on the web portal model.

1.4.5.1 Organisations processing the claim

The claim is processed by two organisations types: the ClaimRepresentative (CR) and the Compensator (COMP); further there is MIB, not detailed in this section.
A single organisation alone is not able to complete the full claim flow: for example a COMP organisation will start processing a claim only when a CR organisation has submitted a new claim to them.

1.4.5.2 Branches of the organisation

Organisations are organised into branches, and users belong to branches.
Branches are usually based on a geographical model or on existing sites/offices/agencies.

1.4.5.2.1 Special branch of COMP organisation

For the COMP organisations, one and only one branch is defined as CentralPoint, which is the default target branch of the AddClaim() command executed by CR users.

1.4.5.3 Profiles processing the claim

The claims percolate to the users through the branches, and are managed in different ways by users with different roles.

1.4.5.3.1 COMP Profiles

The Claim Dispatcher profile (CD) will poll for claims waiting for COMP processing and will allocate the claim to the relevant branch.
The Claim Handler role (CH) will actually enter data or will take decisions on the claim; the CH will receive notifications for timeouts expiring on claims being processed, or actions of the CR user on processed claims.

1.4.5.3.2 CR Profiles

The Claim Handler profile (CH) will actually enter data or will take decisions on the claim; the CH will receive notifications for timeouts expiring on claims being processed, or actions of the COMP user on processed claims.
The CH will poll for claims waiting for CR processing.

1.4.5.4 Administrative console

Every organization will receive a special administrative account able to login onto the administrative console, to create/modify branches, create users with specific profiles.
Please note that only users with "Administrator" profile can login to the administrative console, and such users cannot log in the Settlement section (the claims section of the portal) and in the Reporting section. The organisations Administrator is responsible for maintenance on the account, that the claim handler users have access and those who should no longer be accessing the portal are disabled. Please review the Administrator section of the web site https://www.claimsportal.org.uk/
FROM RELEASE 5 ON: it is possible to create multi-profile users that can access both the Admin Console and the Claims section. This is possible only if the user owns both the profiles of Administrator and Claim Handler.
FROM RELEASE 6 ON: it is possible to assigne the "ADVANCED USER" profile that can access the "Delete" function in order to search and delete claims for GDPR reasons. The profile grants the access to the "Delete" tab only on the Admin Console, i.e. if the profile is assigned to an Administrator user the "Delete" tab will be visible in addition to the other tabs already visible on the Administrator console, if the profile is assigned to an user that is not an Administrator user that user will be able to access the Administrative console but only the "Delete" tab will be visible to the user.

1.4.6 Tailoring the system design to specific needs

In case there is already an existing branch/role design, the actual system usage may depart from the expected system usage according to different models; in this case it is important that careful monitoring on the system is performed in order to detect any issue and coordinate with Crif in case actions are needed.
Some notes follow on nonstandard usage scenarios.

1.4.6.1 Collapsing profiles, branches, users

One single user belonging to a single branch for the organisation.
The system on calling side should track ownership of claims by actual employee.
The system on calling side should track which claims belong to which actual branch.

1.4.6.1.1 Collapsing from CR point of view

The CR will have 1 single branch with 1 single user.
All notifications will be received by the single user, which should collect notifications and delete them from the A2A system as soon as they are no longer needed.

1.4.6.1.2 Collapsing from COMP point of view

The COMP will have 1 single branch, defined as CentralPoint, with 1 single user.
The COMP user will have the CH role; no user will be assigned the CD role.
No need for branch routing of claims: for example no need to call AllocateClaimToBranch(), GetBranchesList().
COMP polling is done by CH user role by calling GetClaimsList()
All notifications will be received by the single user, which should collect notifications and delete them from the A2A system as soon as possible.

1.4.6.2 Collapsing profiles, branches

Multiple users that belong to a single branch of the organisation.
The system on calling side should track which claims belong to which actual branch.

The variation against previous scenario 1.4.6.1, is that notifications are split among multiple users.

1.4.6.3 Collapsing profiles, users

One single user belonging to one of the branches of the organisation.
The system on calling side should track ownership of claims by actual employee.

The variation against previous scenario 1.4.6.1, is that notifications are split among multiple users.

1.4.7 Hints on the error handling

Errors returned by the system can be categorised into two main types of errors:

- **HTTP Error 500 (soap fault)**: in case of unexpected error, the system responds with a soap fault.
- **HTTP Response 200 (ok)**: in case of application failure, the system responds with an error message in the xml response (not a soap fault).

For all error types the system returns a *Trace* that is the error's unique reference on the system's logs. If a call is opened to the HelpDesk the Trace information must be provided so that the Helpdesk are able to collect the relevant information on the issue.

Below are samples of the most frequent errors.

1.4.7.1 HTTP Error 500 (soap fault)

This category of errors occur when:

- the system is not able to process your request or
- the system is not available or
- there is a system error.

As it is not possible to trace back to a specific scope or application's component the "HTTP Error 500" is also defined as "unhandled" error.
For the unhandled errors, due to the unavailability of a specific error message, it is necessary to note down the Trace number and provide it to the HelpDesk if you need support.

Example of a response:

```
<env:Envelope xmlns:env="HTTP://schemas.xmlsoap.org/soap/envelope/">
   <env:Header/>
   <env:Body>
      <env:Fault>
         <faultcode>soap:Server</faultcode>
         <faultstring>Operation failed, Trace:
20220818123440000156934524018O</faultstring>
      </env:Fault>
   </env:Body>
</env:Envelope>
```

Where:

- faultcode: it returns generic information of where the error occurred
- faultstring: it contains the Trace, the error's unique identifier that must be communicated to the HelpDesk

1.4.7.2 HTTP Response 200 (ok)

This category of errors occurs when the system is able to process your request (ok), but there is an issue:

- in the authentication parameters or
- in the format of your request

- in the request itself e.g. the request is performed at a point of the process where it is not possible to perform the action (e.g. try to Acknowlege the liability of a claim where it has already been acknowledged)

Since it is possible to trace back to a specific scope or application's component the "HTTP Response 200" is also defined as "handled" error.
For this type of errors the "Trace" information is available and must be communicated, together with the error message, to the HelpDesk if a call is opened.

We strongly recommend **not** to hard code the wording of the error messages because they may change over time.

Example of responses:

```
<env:Envelope xmlns:env="HTTP://schemas.xmlsoap.org/soap/envelope/">
   <env:Header/>
   <env:Body>
      <ns2:acceptClaimResponse xmlns:ns2="HTTP://wstk.pip.crif.com/">
         <claimInfoResponse>
            <code>Failure</code>
            <message>The operation could not be applied in this point</message>
            <details>Problem Report
       ProcessEngineGuid : DCEA6F60-1EEC-11ED-9789-005056826327
       ActivityEngineGuid: df8fecea-1eec-11ed-9789-005056826327
       Message           : On operation 'acceptClaim_CR19072010' the process is
not freezed in any of the expected activities
==============</details>
            <trace>20220818135740000387969747 1240</trace>
         </claimInfoResponse>
      </ns2:acceptClaimResponse>
   </env:Body>
</env:Envelope>
```

```
<env:Envelope xmlns:env="HTTP://schemas.xmlsoap.org/soap/envelope/">
   <env:Header/>
   <env:Body>
      <ns2:getClaimResponse xmlns:ns2="HTTP://wstk.pip.crif.com/">
         <stringResponse>
            <code>Failure</code>
            <message>ApplicationID Invalid</message>
            <trace>20220818135714000532160707 3361</trace>
         </stringResponse>
      </ns2:getClaimResponse>
   </env:Body>
</env:Envelope>
```

```
<env:Envelope xmlns:env="HTTP://schemas.xmlsoap.org/soap/envelope/">
   <env:Header/>
   <env:Body>
      <ns2:getTokenResponse xmlns:ns2="HTTP://wstk.pip.crif.com/">
         <tokenResponse>
            <code>Failure</code>
            <message>Login Failed or Expired Credential</message>
            <trace>20220818140521000115180203 0470</trace>
         </tokenResponse>
      </ns2:getTokenResponse>
   </env:Body>
</env:Envelope>
```

Where:

- Faultcode: always "Failure"
- Message: contains the information about the error
- Details: optional field that contains additional information about the error, please refer to below table with the indication on when it is returned and when not returned
- Trace: the error's unique identifier that must be communicated to the HelpDesk

1.4.7.2.1 HTTP Response 200 (ok) – most frequent messages

1.4.7.2.1.1 HTTP Response 200 (ok) – login and authentication services

Below are the most frequent messages that you'll find in the responses related to **login and authentication services**.
Please note that for security reasons, the text messages in the responses are generic and the "Details" tag is never returned. In these cases, as per the "HTTP Error 500", it is important to save the Trace number and communicate it to the Help Desk so that it can be used during the investigation on the system.

Error	**Tag "Message"**
Missing/wrong/expired credentials	Login Failed or Expired Credential
Access Token not decriptable	Bad Token
Refresh Token not decriptable	Bad Token
Oru login response failed	Login Failed or Expired Credential
User and UserAsID belong to different Organisations	Login Failed or Expired Credential
Disabled Organisation ID	Login Failed or Expired Credential
Access Token not found	Bad Token
Access Token expired	Bad Token
Access Token invalid	Bad Token
Refresh Token not found	Bad Token
Refresh Token expired	Bad Token
Refresh Token invalid	Bad Token
Input ClientID invalid	Login Failed or Expired Credential
Token generated with UserID = UserAsID and AllowUserEqualsAsUser parameter = null or false	Invalid Token. User and UserAsID cannot be the same for this method

1.4.7.2.1.2 HTTP Response 200 (ok) – Change Password errors

Error	Tag "Message"
Mandatory input parameter missing	Invalid or Expired Credential
Invalid new password	Invalid or Expired Credential
Internal login issue with the old password	Invalid or Expired Credential
Upon first login ever or a password reset, it is mandatory to set your own password.	Please set a new password
You attempted to change ing the password for a user who ich is not an A2A user.	Invalid or Expired Credential

1.4.7.2.1.3 HTTP Response 200 (ok) – process errors

Below is a list of the **most common error messages related to process,** including an indication whether the tag Details are never returned (Not available) or when the details could be returned in cases where additional error information is available on the system (Optional).

Tag "Message"	Tag "Details"
ApplicationID Invalid	Not available
AttachmentGuid Invalid	Not available
Branch ID Invalid	Not available
Claim XML Invalid	Optional
Insurer response XML Invalid	Optional
Input XML not valid	Optional
InputDocumentNotValid	Optional
Claimant representative counter offer XML Invalid	Optional
Compensator counter offer XML Invalid	Optional
You have exceeded the maximum number of attachments for this claim	Optional
Unexpected Null Result	Not available
Error: The attachment exceeds the maximum size limit	Not available
Error: The attachments added to this claim exceeds the total size limit	Not available
The workflow is not freezed in the expected activity	Not available
The process has followed an unexpected path or The process has followed an unexpected path [inner exception message]	Optional
The operation could not be applied at this point	Optional

Organization does not exist	Not available
This function can be called only if the claim is in phase Stage 2 Settlement Pack repudiation	Not available
This function can be called only if the claim is in phase Stage 2 Settlement Pack confirmation	Not available
This function can be called only if the claim is in phase Stage 2 Settlement Pack agreed	Not available
This function can be called only if the claim is in phase 'Liability admitted'	Not available
This function can be called only if the claim is in phase 'Liability admitted (child claim)'	Not available
This function can be called only if the claim is in phase 'Liability admitted with negligence'	Not available
Invalid reference number	Not available
User root organization has invalid 'organization type' business key	Not available
Current User does not have 'COMP Claim Handler Team Leader' profile	Not available
Current User does not have 'CR Team Leader' profile	Not available
The target UserId does not exist	Not available
The target UserId is not valid	Not available
There is a duplicate 'LossType'	Not available
The count of 'LossType' on request/response is not equal	Not available
There is a mismatch between request/response 'LossType'	Not available
Losses must not be present in case of confirmation, repudiation or acceptance of counter offer	Not available
Reached maximum number of requests allowed	Not available
Unable to execute operation, 'UserAsID' is not a Team Leader	Not available
Process information N/A	Not available
The reason for extension is not valid	Not available
Interim Timeout already extended	Not available
Invalid Interim Timeout request	Not available
Invalid Alternative Vehicle Provision	Not available
Insufficient Privileges to read document with PrintableDocumentGUID + document GUID	Not available
Insufficient Privileges to read document with PrintableDocumentGUID + document GUID	Not available
No Insurer Central Point found for organization + org short description	Not available
Login Failed or Expired Credential	Not available

Bad Token	Not available
Invalid or Expired Credential	Not available
Invalid Token. User and UserAsID cannot be the same for this method	Not available
Operation Failed	Generic error message. Details tag is optional.
ProcessEnd	Optional

If you are still unable to correctly interpret your issue, please contact the Help Desk and include in your email:

- the Trace number
- the Error message
- the content of the Details node if present.
- the full XML request, please note that in case of live request containing live data the request must be password protected.

1.4.8 Maintaining updated info within the in-house CMS

Claims Portal is the media where claims are processed by involved parties, and contains real-time updated info.
If an organisation is using a CMS which interacts with Claims Portal, an important point is to ensure that the CMS is working on up-to-date info.
This means that some kind of refreshing or polling should be done, to detect when relevant info is changed (this means there is an advance in the claim processing).
With regards to the information polling there are some aspects:

- How 'changed' info is detected
- Frequency of changed info detection
- Which parts of the claim changes when

These points will be cleared in the following, together with other relevant points.

1.4.8.1 Recap of parts of a claim

A claim consists of the following parts:

- CNF values (always retrieved with GetClaim, regardless of the stage)
- CNF processing status (retrieved with GetClaimsStatus)
- Attachments (retrieved with GetAttachmentsList)
- PDF of the CNF (retrieved with GetPrintableDocumentsList)

1.4.8.2 Changes: about attachments

Please note that attachments can be added when the claim is with the other party, however a check for new attachments must be made when the claim changes ownership. To check for new attachments whilst the claim is with the other party, refer to Identify New Attachments in the Instructions for Developers Section. Continuous submission of the SearchClaim or the GetAttachmentsList call is not permitted.

1.4.8.3 Changes: about printable documents

Updated printable documents are only generated when the claim changes ownership.

1.4.8.4 Changes: the role of notifications

Notifications are meant to notify one side when something relevant happened on the other side. For example, when the COMP states that Article75 applies, a notification is sent to the CR, with the purpose to make the CR aware that the timeout has changed.

Please note that notifications are not generated when the claim changes ownership: i.e. when the claim is sent from the CR to the COMP and viceversa, no notification is generated.

1.4.8.5 How changed info is detected (=detecting claims waiting for processing)

The underlying proposition is that a given claim should be processed only when it is no longer under the ownership of the other side, and this happens when the claim moves on specific phases.
Consequently there is no need to monitor any claim change while the claim is under the ownership of the other side, and has not reached the phases relevant for the other side.

To detect which claims have reached a given phase, use the command SearchClaims(), specifying as criteria the phase ID.
To cover the whole life of the claim, multiple SearchClaims() should be executed to cover all relevant phases.

To retrieve the list of relevant phases, depending on whether you are a CR or COMP, please examine the workflows in section 1.3, and based on this go to the details of the commands in section 2.

1.4.8.6 Deprecated detection of changed info

Detecting any changed info, by a massive re-load of all parts of all claims currently in progress in the CMS: i.e. for each claim currently in the CMS, executing GetClaim, GetClaimsStatus, GetAttachmentsList, GetPrintableDocumentsList should not be used.
This approach should not be adopted as:

1) A lot of A2A calls are executed (4 times the number of claims), requiring time (and loading the A2A system)
2) A lot of processing is required to detect what actually has changed
3) The CMS is loaded with a lot of updates

Hence a massive re-load of all parts of all claims should not be attempted.
The A2A system monitoring is able to detect this approach and will react accordingly, slowing or stopping the account performing this A2A behaviour.

1.4.8.7 Frequency of changed info detection

This greatly depends on the number of claims being processed, the number of organisations processing the claims, the reaction time of such organisations.
According to our estimation, a reasonable balance is to check for info changes every hour.

1.4.8.8 Status changes in in-house CMS are not tracked within Claims Portal

If the in-house CMS manages the status of claims, please always bear in mind that CMS statuses may not match Claims Portal phases, and Claims Portal won't track the status of claims within the CMS, but only the phases according to the Claims Portal flow.

For example, when a claim is created by a CR, is in phase "Claim Submitted"; if the CMS

of the COMP detects this with SearchClaims(criteria on phaseID ClaimSubmitted) and loads the CNF data within the CMS, this means that this claim is in a status of "LoadedIntoCMS" which is different from the status "YetToBeLoadedIntoCMS" of other claims: this different status is not tracked by Claims Portal, which is not aware of this:
the consequence is that when executing SearchClaims(criteria on phaseID ClaimSubmitted), Claims Portal will return claims regardless of claims being already loaded in the CMS or not: Claims Portal will not return only "YetToBeLoadedIntoCMS" claims", because this info is only tracked into the CMS: so the software communicating with Claims Portal may need to take care of this status which make sense only in the CMS but not on Claims Portal, and this is something that could be done only on the client side, because it is totally dependent on the way the CMS works.

1.4.9 Notes on the communication with the A2A server

1.4.9.1 HTTPS protocol

Claims Portal A2A server is configured to work in the HTTPS protocol, which is an industry standard secure communication protocol to exchange commands and information.

1.4.9.2 Default communication timeout

Please note that the processing of an A2A command may require some time, so the client should be configured with a timeout long enough to wait for the reply from the server.
In case a timeout is experienced, please be aware that this only means the client stopped listening for the server response, but actually the server is still processing the command up to its completion, so a subsequent call to GetClaimsStatus() will confirm that the process has moved to the following step in the flow.
Please bear in mind that company network devices may monitor the communications throughout the company network, and there may have special configuration to detect and close communications which appear to be idle: so in case network connection is closed despite a long client timeout configuration, network devices in the path to the A2A server should be investigated.

1.4.9.3 Network devices (company firewall, etc)

If the A2A server URL is not reachable, ensure relevant network devices in the middle to the network path to the A2A server are correctly configured to allow network traffic to and from the A2A server.
For example, firewall may require a special configuration to open a port to the IP address of the A2A server, depending on the protocols allowed by the company network policy.
Further, the firewall may require a specific configuration to allow access to the HTTPS protocol.

1.4.10 Client implementation

1.4.10.1 Familiarize with the process using the WEB portal

Before implementing the client, it is important to be familiar with the claim workflow using the WEB portal: using the forms visualization allows a better comprehension of the flow, required data and output of the system.

1.4.10.2 Implement the client to allow retrieving XML Request and XML Response

The client should be implemented to allow for retrieval of the XML Request and the XML Response.

1.4.10.3 Implement the client against TEST environment

The implementation of the client should be finalized by sending commands to the Claims Portal system available in TEST environment. All relevant commands should be covered, and a number of different scenarios should be covered.
CRIF is always available to help covering any functionality or scenario, to ensure that the client implementation is ready to communicate with the Live environment.

1.4.11 Troubleshooting

1.4.11.1 Identify if the problem is in client implementation

When the A2A client implementation is not completely compliant with the documentation, it may not be able to correctly communicate with the Claims Portal A2A system, so errors may be experienced.
Not all problems could be due to the Claims Portal system A2A, so it's important to identify if the error may be related to the client implementation.
For example errors with text "Object not found" or "Object is null" are easily caused by class wrappers in the client implementation.

1.4.11.2 Reproduce the issue on the WEB portal

To ensure the problem is not caused by a misunderstanding of the documentation, attempt to replicate the issue with the WEB portal.

1.4.11.3 Do not submit a query without providing XML Request and XML Response

When submitting a query to the Claims Portal HelpDesk; always include the XML Request (and Claim ID) and the XML Response.
This allows us to exactly understand; which is the command, the parameters, the accounts involved, and allows the HelpDesk to start investigating; unless passwords are specifically requested, you may omit or replace with special characters to maintain account security.

This means that the A2A client software should be designed to collect - for troubleshooting purposes - the XML Request and XML Response communicated with Claims Portal.
The XML Request, together with the ClaimID allows analysis of the claim life, and allows us to reproduce the issue in the CRIF laboratory.

If the client is unable to provide the troubleshooting info, the user must provide as a minimum the following information:

1) Command name
2) Claim ID
3) A2A account
4) UserAsID account
5) Data relevant for the command call

Please note that if the troubleshooting information is not provided the time taken for the investigation will be longer and we cannot guarantee that we can assist the user with their query. Always provide the **XML Request (and Claim ID) and the XML Response.**

1.4.11.4 Allow A2A client design for out-of-the-process or web operations

The comprehension of the system requires some time to be fully understood; consequently, the client should be designed to be able to allow one or more steps to be completed out of the process (for example if a specific operation requires a workaround, or should be done on the web portal).

1.4.12 Applicable timeouts during the claim flow

Here is a recap of timeouts applicable in the claim flow (in 2 different scenarios: since the flow changes based on the scenario, there are many more examples).
To allow CMS to retrieve the timeout in use in the Claims Portal system, the table shows the relevant timeout in term of the TimeOutType and SelectedTimeOut elements returned by GetClaim command (AFTER a given command is executed).

Scenario 1

Who	*Executed command*	*TimeOutType=*	*SelectedTimeOut=*
CR	AddClaim		//Insurer/@LiabilityDecisionTimeout
COMP	ClaimAcknowledgement		//Insurer/@LiabilityDecisionTimeout
COMP	AcceptClaim		//Insurer/@LiabilityDecisionTimeout
COMP	SendLiabilityDecision	<not relevant>	<not relevant>
CR	AcknowledgeLiability	<not relevant>	<not relevant>
CR	SetInterimPaymentNeeded (true)	<not relevant>	<not relevant>
CR	AddInterimSPFRequest(greater£1000)	21	//InterimSettlementPack/HighPayment/@InterimPaymentTimeout
COMP	AddInterimSPFResponse	21	//InterimSettlementPack/HighPayment/@InterimPaymentTimeout
CR	SetStage21Payment	<not relevant>	<not relevant>
CR	AddS2SPFRequest	221	//Stage2SettlementPack/Stage2DecisionOrCounterOffer/@Stage2DecisionOrCounterOfferTimeout
COMP	AddS2SPFResponse	222	//Stage2SettlementPack/Stage2DecisionOrCounterOffer/@Stage2DecisionOrCounterOfferTimeout
CR	SetStage2SPFCounterOfferNeeded(true)	222	//Stage2SettlementPack/Stage2DecisionOrCounterOffer/@Stage2DecisionOrCounterOfferTimeout
CR	AddStage2SPFCounterOfferByCR	222	//Stage2SettlementPack/Stage2DecisionOrCounterOffer/@Stage2DecisionOrCounterOfferTimeout
COMP	AddStage2SPFCounterOfferByCM	222	//Stage2SettlementPack/Stage2DecisionOrCounterOffer/@Stage2DecisionOrCounterOfferTimeout
CR	SetStage2SPFCounterOfferNeeded(false)	222	//Stage2SettlementPack/Stage2DecisionOrCounterOffer/@Stage2DecisionOrCounterOfferTimeout
CR	SetStage2SPFAgreementDecision(false)	<not relevant>	<not relevant>
CR	AddCPPFRequest	f370	//CourtProceedingsPack/CourtProceedings/@CourtProceedingsTimeout
COMP	AddCPPFResponse	<not relevant>	<not relevant>

Scenario 2: *ExtendInterimPaymentDecisionTimeout, ExtendStage2SPFDecisionTimeout,ExtendStage2SPFCounterOfferTimeout*

Who	*Executed command*	*TimeOutType=*	*SelectedTimeOut=*
CR	AddClaim		//Insurer/@LiabilityDecisionTimeout
COMP	ClaimAcknowledgement		//Insurer/@LiabilityDecisionTimeout
COMP	AcceptClaim		//Insurer/@LiabilityDecisionTimeout
COMP	SendLiabilityDecision	<not relevant>	<not relevant>
CR	AcknowledgeLiability	<not relevant>	<not relevant>
CR	SetInterimPaymentNeeded (true)	<not relevant>	<not relevant>
CR	AddInterimSPFRequest(greater£1000)	21	//InterimSettlementPack/LowPayment/@InterimPaymentTimeout
COMP	ExtendInterimPaymentDecisionTimeout	21ex	//InterimSettlementPack/LowPayment/@InterimPaymentTimeout
COMP	AddInterimSPFResponse	21ex	//InterimSettlementPack/LowPayment/@InterimPaymentTimeout
CR	SetStage21Payment	<not relevant>	<not relevant>
CR	AddS2SPFRequest	221	//Stage2SettlementPack/Stage2DecisionOrCounterOffer/@Stage2DecisionOrCounterOfferTimeout
COMP	ExtendStage2SPFDecisionTimeout	22ex	//Stage2SettlementPack/Stage2DecisionOrCounterOffer/@Stage2DecisionOrCounterOfferTimeout
COMP	AddS2SPFResponse	222	//Stage2SettlementPack/Stage2DecisionOrCounterOffer/@Stage2DecisionOrCounterOfferTimeout
CR	SetStage2SPFCounterOfferNeeded(true)	222	//Stage2SettlementPack/Stage2DecisionOrCounterOffer/@Stage2DecisionOrCounterOfferTimeout
CR	ExtendStage2SPFCounterOfferTimeout	22ex	//Stage2SettlementPack/Stage2DecisionOrCounterOffer/@Stage2DecisionOrCounterOfferTimeout
CR	AddStage2SPFCounterOfferByCR	22ex	//Stage2SettlementPack/Stage2DecisionOrCounterOffer/@Stage2DecisionOrCounterOfferTimeout
COMP	ExtendStage2SPFCounterOfferTimeout	22ex	//Stage2SettlementPack/Stage2DecisionOrCounterOffer/@Stage2DecisionOrCounterOfferTimeout
COMP	AddStage2SPFCounterOfferByCM	22ex	//Stage2SettlementPack/Stage2DecisionOrCounterOffer/@Stage2DecisionOrCounterOfferTimeout
CR	SetStage2SPFCounterOfferNeeded(false)	222	//Stage2SettlementPack/Stage2DecisionOrCounterOffer/@Stage2DecisionOrCounterOfferTimeout
CR	SetStage2SPFAgreementDecision(false)	<not relevant>	<not relevant>
CR	AddCPPFRequest	f370	//CourtProceedingsPack/CourtProceedings/@CourtProceedingsTimeout
COMP	AddCPPFResponse	<not relevant>	<not relevant>

1.4.13 Phases

Here is a listing of phases applicable in the claim flow; the listing is in chronological order, but of course not all can be reached for any claim, since this depends on the claim options selected.

Phase name	*Compatibility level (RELEASES)*	*Ownership on such phase*

ClaimDataInputEdit	0	CR
ClaimSubmitted (Claim submitted)	0	COMP
ClaimRejectedToCR	0	CR
AcknowledgeRejectedClaim	5	CR
ClaimRejectedEnd	5	CR
LiabilityDecision (Liability decision)	0	COMP
LiabilityAdmitted (Liability admitted)	0	CR
LiabilityRejected (Liability not admitted)	0	CR
LiabilityAdmittedWithNegligence (Liability admitted with negligence)	0	CR
LiabilityAdmittedChildClaim (Liability admitted (child claim))	0	CR
LiabilityDecisionTimeout	0	CR
FraudStated (Fraud stated)	0	CR
Stage1Complete	0	
StartOfStage21 (Start of stage 2.1)	0	CR
InterimSettlementPackForm (Interim settlement Pack Form)	0	CR
InterimPaymentDecision (Interim Payment decision)	0	COMP
InterimPaymentDecisionTimeout (Interim Payment Decision Timeout)	0	CR
InterimPaymentDecisionTimeoutEnd (Interim Payment Decision Timeout End)	0	
InterimSettlementPackRejected (Interim Settlement Pack rejected)	0	CR
AcceptanceOfPartialInterimPayment (Acceptance of partial Interim payment)	0	CR
DecisionForPartialInterimPayment (Decision for partial Interim payment)	0	CR
WaitingForInterimPayment (Waiting for Interim payment)	0	
Stage21Complete (Stage 2.1 complete)	0	
Stage2SettlementPackForm (Stage 2 Settlement Pack Form)	0	CR
Stage2SettlementPackDecision (Stage 2 Settlement Pack decision)	0	COMP
Stage2SettlementPackDecisionTimeout (Stage 2 Settlement Pack decision Timeout)	0	CR
Stage2SettlementPackDecisionTimeoutEnd (Stage 2 Settlement Pack decision Timeout End)	0	
Stage2SettlementPackRepudiation (Stage 2 Settlement Pack repudiation)	0	CR
Stage2SettlementPackRepudiationEnd (Stage 2 Settlement Pack repudiation End)	0	
Stage2SettlementPackConfirmation (Stage 2 Settlement Pack confirmation)	0	CR
Stage2SettlementPackConfirmationEnd (Stage 2 Settlement Pack confirmation End)	0	
Stage2SettlementPackCounterOffer (Stage 2 Settlement Pack counter offer)	0	CR
Stage2SettlementPackCounterOfferByCM (Stage 2 Settlement Pack counter offer (Compensator)	0	COMP
Stage2SettlementPackCounterOfferDecision (Stage 2 Settlement Pack counter offer decision)	0	CR
Stage2SettlementPackAgreementDecisionComp (Stage 2 Settlement Pack agreement decision (Compensator))	0	COMP

Stage2SettlementPackAgreementDecision (Stage 2 Settlement Pack agreement decision)	0	CR
Stage2SettlementPackAgreed (Stage 2 Settlement Pack agreed)	0	COMP
Stage2SettlementPackAgreedCR (Stage 2 Settlement Pack agreed (CR))	0	CR
Stage2SettlementPackNotAgreedCR (Stage 2 Settlement Pack not agreed (CR))	0	CR
Stage2SettlementPackAgreedEnd (Stage 2 Settlement Pack agreed End)	0	
CourtProceedingsPackForm (Court Proceedings Pack Form)	0	CR
CourtProceedingsPackFormResponse	0	COMP
EndOfStage2	0	
ExitProcess (Exit Process)	0	CR, COMP
ExitProcessEnd (Exit Process End)	0	CR, COMP

1.4.14 Interface Compatibility level

1.4.14.1 Definition

The compatibility level is used to indicate which A2A interface elements are accessible by a client depending on both the release in place and the version of each claim stored in the portal (please note the workflow process versions reported below are for explanation only).

Range of workflow process version	Compatibility level (RELEASES)	Notes
Any	All	This means is compatible with every level (used usually for commands which do not affect the workflow)
1.0 ...	0	Release 0 on 01/04/2013

1.4.14.2 How to read the Compatibility Level information

The following table explains how to read the info about the Compatibility level:

Reported compatibility level	Meaning
All	This command can be executed regardless of the claim version (usually relates to commands which do not affect the workflow)
0	This command can be executed only on claims belonging to a workflow process version in the range 1.0 ... 2.0
1	This command can be executed only on claims belonging to a workflow process version in the range 2.1 ...
0, 1	This command can be executed on claims belonging to a workflow process version either in range 1.0 ... 2.0 or in range 2.1 ...
0*, 1	This command can be executed on claims belonging to a workflow process version either in range 1.0 ... 2.0 or in range 2.1 ... The star highlights that the command signature has changed (some arguments have been added) or the XML of an argument has changed its XSD.

1.4.14.3 Process version used to switch between releases

Please note that in the above section, the version numbers 2.0 and 2.1 are just examples.
The first process version implemented for a new Release will be communicated in advance to all relevant audience at the latest two weeks before the deployment in the Production environment.
For more info on how the process version will be used together with the command GetSystemProcessVersion, see details in 2.2.40

1.5 Timeout values of the Test site

The following table shows the reduced timeout values of the Test site.

The timeout values for the Production environment are shown as well in order to provide an overview of the standard timeout values used in the live site.

In order to correctly read the timeout values for the Test site, please note that:

- All the timeout values are expressed in Business Days;
- Timeout values are intended as relative timeouts. For example, the Stage 2 Settlement pack decision timeout is calculated starting from the business day after the date sent of the Stage 2 Settlement Pack Request;
- All warnings (1 day left, 3 days left, etc.) are triggered instantaneously; therefore they are not coherent with the reduced timeout values.

Workflow Section	Phase Description	Timeout for PRODUCTION	Timeout for TEST SITE
STAGE 1	Liability decision timeout (EL/ELD claim)	30	1
	Liability decision timeout (PL claim)	40	2
STAGE 2.1	First Interim payment decision timeout (request for £1,000)	10	1
	First Interim payment decision timeout (request for more than £1,000)	15	2
	Subsequent Interim payment decision timeout (request for £1,000 or more)	15	2
	Extend time for CRU, only if payment is greater than £1,000	+15	+1
STAGE 2.2	Stage 1 costs payment timeout (used only as a reminder)	10	1
	Stage 2 Settlement pack decision timeout	15	1
	Stage 2 Settlement pack counter offer decision timeout	+20	+1
	Stage 2 Settlement pack automatic extension	+5	+5
STAGE 3 COURT PROCEEDINGS	Court Proceedings pack response timeout (used only as a reminder)	5	5

1.6 Process versions of the Test site

The following table shows the process versions of the Test site.

The process versions for the Production environment are shown as well in order to provide an overview of the actual versions used in the live site.

In order to correctly read the values in the table below, please note that the range provided is intended limits included, e.g. the range from v1.0 to v3.4 will contain all the applications with process version from v1.0 inclusive to v3.4 inclusive.

Environment	From	To	Release version
TEST SITE	1.0	1.9	Release 3
	2.0	2.2	Release 4*
	2.3	-	Release 5
	6.0		Release 6
PRODUCTION	1.0	1.6	Release 3
	1.7	1.8	Release 4*
	5.0	-	Release 5**
	6.0		Release 6

*Note: Release 4 did not introduce any changes to A2A interface. Schemas and functions valid for Release 3 are valid also for Release 4 claims.

**Note: Release 5 will start from the process version “5”. All the process versions from 4.2 to 4.9 have never existed.

2. Specific Requirements

2.1 Authentication

2.1.1 Renamed input fields

Please note that the following existing input field have been renamed as follows:

- username > userID
- asUser > userAsID

Moreover, the following field, introduced with A2A token (specs v5.2) have been renamed as follows:

- AccessToken > accessToken
- RefreshToken > refreshToken
- AccessTokenExpiresIn > accessTokenExpiresIn
- RefreshTokenExpiresIn > refreshTokenExpiresIn

In addition, a new namespace:

From *xmlns:ws=http://ws.elpl.crif.com/* to *xmlns:wstk=http://wstk.elpl.crif.com/*

In place of "ws" the system now uses "wstk".

2.1.2 GetToken

2.1.2.1 GetToken

This request provides the functionality to get an access token that must be used in all the business functionalities. The username and password are now used to request a token.

INPUT:

- userID – username for the A2A
- password – password for the A2A
- userAsID – username of the user in Claims Portal

- clientID – The clientId is an attribute that will be used to uniquely identify the software house that is performing the getToken() or the refreshToken() on behalf of the COMP/CR. The allowed format is SHnnnn (SH + four numbers). If you are a software house and you have not been allocated a clientID, you must obtain one from ClaimsPortal@mib.org.uk

2.1.2.2 GetTokenResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When it is successful, the system responds with:

- accessToken
- refreshToken
- accessTokenExpiresIn
- refreshTokenExpiresIn

accessToken is valid for 7200sec (2 hours);
refreshToken is valid for 86400sec (24 hours);

2.1.2.3 Error handling specific notes

See section 1.4.7.2.1.1 for login and authentication specific error messages.

2.1.3 RefreshToken

2.1.3.1 RefreshToken

This request provides the functionality to get a new access token valid for 2 hours.

INPUT:

- userID – username for the A2A
- refreshToken – refreshToken
- userAsID – username of the user in Claims Portal
- clientId – The clientId is an optional attribute that will be used to uniquely identify the software house that is performing the getToken() or the refreshToken() on behalf of the COMP/CR. The allowed format is SHnnnn (SH + four numbers). If you are a software house and you have not been allocated a clientID, you must obtain one from ClaimsPortal@mib.org.uk

2.1.3.2 RefreshTokenResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When it is successful, the system responds with:

- AccessToken
- RefreshToken
- accessTokenExpiresIn
- refreshTokenExpiresIn

- AccessToken is valid for 7200sec (2 hours);
- RefreshToken is valid for 86400sec (24 hours) minus the time passed for first RefreshToken valid provided;

2.1.3.3 Error Handling specific notes

See section 1.4.7.2.1.1 for login and authentication specific error messages.

2.1.4 ChangePassword

2.1.4.1 PasswordChange

This request provides the functionality to set A2A user's new password.

INPUT:

- userID – username for the A2A
- oldpassword – oldpassword for the A2A
- newpassword – newpassword for the A2A

2.1.4.2 PasswordChangeResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When it is successful, the system responds with:

- passwordChangeResult

2.1.4.2.1 Password requirements

For security purposes the password must follow the rules below:

- The password must be at least 12 characters long
- It is not possible to use as password the user id or name or surname or organisation id
- The password must contain at least one number, one capital letter and one symbol
- Allowed symbols: |!"£$€%&/=?^'*+@°#,;:.-_()
- The password must not contain the word "password" and none of its permutations (e.g. "pwd", "pswd"…) both in capital and/or lowercase letters
- It is not possible to reuse ANY old password

Please note that the system retains a certain number of old passwords; this functionality is called password history. Setting of password history is not disclosed for security reasons and may vary based on the environment.

Minimum length of password: 12 chars.
Maximum length of password: not to exceed 32 chars.

2.1.4.2.2 Specific notes for this mehtod

Below the main specific notes and behaviors for this method:

- For security reasons, after a certain number of failed attempts the user is blocked and must contact the Help Desk for support..
- It is possible to change the password via the changePassword() method for A2A users only.
- Upon first login or a password reset, it is mandatory to set your own password; the message "Please set a new password" will appear.

2.1.4.3 Error handling specific notes

See section 1.4.7.2.1.2 for change password specific error messages.

2.2 Functionalities

2.2.1 AddClaim

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-1	**Add Claim**	**High**	**All**

System status before execution:

Can be executed by:	**CR**
Phase applicable:	*<not applicable here>*

System status after execution:

Phase reached:	**Claim Submitted**

2.2.1.1 AddClaim

This request provides the functionality to add a new claim into the system.
The claim is automatically routed to the CentralPoint branch of the target Compensator.
The NotificationDate used for timeouts must be provided as input and will be the date of the claim submission.

INPUT:

- accessToken – accessToken for the A2A
- claimXML – the content of the claim notification form (see schema DocumentInput.xsd).

2.2.1.2 AddClaimResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

In case of success, the system responds with:

- claimInfoResponse
  - code – ok
  - claimInfo
    - activityEngineGuid
    - applicationId
    - phaseCacheId
    - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.1.3 Error Handling specific notes

See section 1.4.7.2.1 for a list of possible errors returned by this functionality.
Below are the most frequent errors specific to the AddClaim method:

```
Case "InputDocumentNotValid": => XML to be re-built (validate XML against XSD for wrong node)
Case "ProcessEnd": => XML to be re-edited (inspect ErrorData for business error details)
```

`Case "InsufficientPrivileges": =>` Authentication problem (CMS account management issue?)

If you are still not able to correctly interpret your issue, please contact the Help Desk and include in your email:

- the Trace number
- the Error message
- the content of the Details node if present.
- the XML request, please note that in case of live request containing live data the request must be password protected.

2.2.2 Get claims list

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-2	**GetClaimsList**	**High**	**All**

2.2.2.1 GetClaimsList

This request provides the functionality to get the information found on the work list of a single user. The data structure returned includes all the information needed for retrieving details on a single claim and its status.

INPUT:

- accessToken – accessToken for the A2A

2.2.2.2 GetClaimsListResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When it is successful, the system responds with:

- claimsListResponse
  - Code – ok
  - claimsList
    the list of claims belonging to the worklist of the user "userAsID". Per each item:
    - activityEngineGuid
    - attachmentsCount
    - applicationId
    - applicationReferences – [the CR Reference Number and the Defendant's Insurer (alias CM) Reference Number]
    - attachmentsCount
    - claimType
    - creationTime
    - currentUserID
    - gdprClaimDeleted FROM RELEASE 6 ON. Please note this is only a change on the XSD, if gdprClaimDeleted=1 the claim will not appear in the retrieved list
    - lockStatus – possible values:
      1 = Not locked
      2 = locked by a user different from the one who performed the GetClaims

3 = locked by THE user who performed the GetClaims
- lockUserId
- phaseCacheId
- phaseCacheName
- printableDocumentsCount
- versionMajor
- versionMinor

In case of failure or error, see section 1.4.7

2.2.3 Search claims

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-3	**SearchClaims**	**High**	**All**

2.2.3.1 SearchClaims

This request provides the functionality to search for a claim by specifying a set of search criteria.
Search criteria can be specified on: claimID, Claim CR Reference number, Claim CM Reference Number, phase, branch, etc.

INPUT:
- accessToken – accessToken for the A2A
- searchClaimCriteria
  - applicationID
  - claimType
  - branchPath*
  - CMReferenceNumber *(Claim CM Reference number)*
  - CRReferenceNumber *(Claim CR Reference Number)*
  - dispatchDateFrom**
  - dispatchDateTo**
  - phaseCacheId
  - sentDateFrom *(date when claim was submitted to Comp, in phase 'ClaimSubmitted')*
  - sentDateTo
  - sortField *(see possible values in WSDL)*
  - sortOrder *(see possible values in WSDL)*
  - startDateFrom *(date of claim editing start, in phase 'ClaimDataInputEdit')*
  - startDateTo

*=filter available to Claimant Representatives only

**=defined in WSDL for future use

2.2.3.2 SearchClaimsResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:
- claimInfosListResponse
  - Code – ok
  - claimsList

the list of claims belonging matching the given criteria and visible to the user "userAsID". Per each item:

- activityEngineGuid
- applicationId
- applicationReferences
- attachmentsCount
- claimType
- creationTime
- currentUserID
- FROM RELEASE 6 ON: gdprClaimDeleted*
- lockStatus – possible values:
  1 = Not locked
  2 = locked by a user different from the one who performed the SearchClaims
  3 = locked by THE user who performed the SearchClaims
- lockUserId
- phaseCacheId
- phaseCacheName
- printableDocumentsCount
- versionMajor
- versionMinor

*Field present and set to 1 onlyif the claim has been deleted for GDPR reasons.

Please note that the search output is limited in size for performance reasons to a maximum of 5000 records. If the searchClaims() retrieves more than 5000 records, it would generally mean that the client is not using the additional filters available in order to refine the search, e.g. filtering by "phaseCacheId", "startDate", "sentDate", etc.

Note: in order to optimise the response times for each SearchClaims() call, you should always use the additional search filters e.g. filtering by "phaseCacheId", "startDate" or "sentDate"

In case of failure or error, see section 1.4.7

2.2.4 Get claim

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-4	**GetClaim**	**High**	**All**

2.2.4.1 GetClaim

This request provides the functionality to get all the information stored within a particular claim up to the phase in which the claim has arrived. The data returned by the functionality includes process and business information.

INPUT:

- accessToken – accessToken for the A2A
- applicationId

2.2.4.2 GetClaimResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- stringResponse
  - code – ok
  - value – this node contains the xml of the requested claim. The structure of this xml is described in the schema**.**

In case of failure or error, see section 1.4.7

2.2.5 Get claim status

The functionality is used to retrieve the status of a particular claim. This function can be used in order to know the position in the process of a particular claim and to update the claim using the right command.

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-5	**GetClaimStatus**	**High**	**All**

2.2.5.1 GetClaimStatus

This request provides the functionality to retrieve the status of a particular claim. This function can be used in order to know the position in the process of a particular claim and to update the claim using the right activityEngineGUID.

INPUT:

- accessToken – accessToken for the A2A
- applicationId

2.2.5.2 GetClaimStatusResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- claimInfo2Response (FROM RELEASE 6 ON)
  - code – ok
  - claimInfo
    - activityEngineGuid
    - applicationId
    - phaseCacheId
    - phaseCacheName
  - FROM RELEASE 6 ON: gdprClaimDeleted*

*Field present with attribute ClaimDeleted equal to 1 only if the claim has been deleted for GDPR reasons.
Please note that the new ClaimInfo2Response node is a replacement for the old ClaimInfoResponse node and it will be returned also in case of getClaimStatus() of claims added with a previous release.

In case of failure or error, see section 1.4.7

NOTE: Please note that in order to obtain the status information for several claims with a single call you should use the SearchClaims(). See the SearchClaims() details for info on the additional info in section 2.2.3.

2.2.6 Get Organisation

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-6	**GetOrganisation**	**High**	**All**

2.2.6.1 Get**Organisation**

This request provides the functionality to get the details of an Organisation starting from the path.

INPUT:

- accessToken – accessToken for the A2A
- organisationPath – the Organisation Path (=”/”+<OrganisationId>) to be get.

2.2.6.2 Get**Organisation**Response

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- code – ok
- organisation
    - organisationId
    - organisationName
    - organisationPath
    - address (it’s a set of fields)

In case of failure or error, see section 1.4.7

2.2.7 Get branches list

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-7	**GetBranchesList**	**High**	**All**

2.2.7.1 GetBranchesList

This request provides the functionality to get the list of the branches of the Organisation of the user "userAsID".

Since the list of branches does not change frequently, it should be called only in case of reasonable doubts that the list was updated after the last time the same function was called.
The A2A clients should cache this info.
The misuse of this feature could impact the performance of the system.

INPUT:

- accessToken – accessToken for the A2A

2.2.7.2 GetBranchesListResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- Code – ok
- organisationInfosList
  the list of branches of the organisation (together with the info about their status enabled/disabled)
  For each item:
  - organisationId
  - organisationName
  - organisationPath

In case of failure or error, see section 1.4.7

2.2.8 Get hospitals list

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-8	**GetHospitalsList**	**High**	**All**

2.2.8.1 GetHospitalsList

This request provides the functionality to perform a search within the list of NHS Hospitals in the system, given some search criteria. This is a search function, it is not possible to download the full list and this should not be attempted.

The misuse of this feature could impact the performances of the system.

INPUT:

- accessToken – accessToken for the A2A
- hospitalName – text to be searched in the hospital name
- postCode – postcode to be searched

2.2.8.2 GetHospitalsListResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- code – ok
- hospitalsList
  the list of public Hospitals that matches with the input criteria
  For each item:
  - name
  - postCode
  - addressLine1
  - addressLine2
  - addressLine3
  - addressLine4

In case of failure or error, see section 1.4.7

2.2.9 Get notifications list

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-9	**GetNotificationsList**	**High**	**All**

2.2.9.1 GetNotificationsList

This request provides the functionality to get the list of the Notifications available to the user "userAsID" (and displayed to this user in the section "Notifications" of the webUI) not yet deleted from the system.

This function must be called periodically by the A2A client to keep their system up to date.
The misuse of this feature could impact the performances of the system.

INPUT:

- accessToken – accessToken for the A2A

2.2.9.2 GetNotificationsListResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- code – ok
- notificationsList
  the list of the Notifications not yet deleted from the system
  For each item:
  - formattedDate
  - notificationDateTime
  - notificationGuid
  - notificationMessage
  - applicationId

In case of failure or error, see section 1.4.7

FROM RELEASE 5 ON:
The bulk transfer functionality allows an organisation to transfer claims to another organisation of the same type (CR to CR, COMP to COMP).
Every claim is always handled by a CR (i.e. CR1) and a COMP (COMP1). When a claim is transferred from CR1 to CR2, the third party organisation (COMP1) is indirectly involved in the transfer and must be advised that the other party is changing. Similarly, when a claim is transferred from COMP1 to COMP2, the third party organisation (CR1) is indirectly involved in the transfer and must be advised that the other party is changing.
In order to inform the third party organisations involved in a bulk transfer process, new notifications are generated and sent only to their A2A. From Release 5 on, the getNotificationList request provides the functionality to get the list of the Notifications available to the A2A user that have not yet been deleted from the system.

FROM GetToken() RELEASE ON:

As the notifications belong to the A2A user the getNotificationList() request must be sent using an ad hoc generated token where User = UserAsID.
In order to avoid receiving an error when calling the getNotificationList() using the token generated with User = UserAsID the parameter "A2Anotification" **must be set to TRUE**.
If the parameter is set to FALSE or the parameter is NOT INCLUDED in the request the system will return the following error "Invalid Token. User and UserAsID cannot be the same for this method", see paragraph 1.4.7 for further details.

Please note that this is valid only for the getNotificationsList() and the **removeNotification()** for all other calls the provided token generated with the User = UserAsID will return an error.

This function must be called on a daily basis by the A2A client to keep their system up to date.

INPUT:

- accessToken – accessToken for the A2A
- A2ANotification - flag to extract the A2A notifications. Must be set to "true" to download the A2A notifications. Do not pass the parameter or set it to "false" to download web browser notifications.

FROM RELEASE 6 ON a new notification has been introduced. This notification will inform the users that a claim has been removed from the system due to GDPR reasons. Both the COMP and the CR users that have worked on the claim will receive the notification.
To retireve the notification the same instructions explained for the Bulk Transfer notification **MUST** be followed.

2.2.10 Remove notification

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-10	**RemoveNotification**	**High**	**All**

2.2.10.1 RemoveNotification

This request provides the functionality to delete the notification from the system.

NOTE → Depending on the user that has been used to obtain the accessToken the deletion of a Notification can be possible or not.
INPUT:

- accessToken – accessToken for the A2A
- notificationID – the id of the notification to be deleted

2.2.10.2 RemoveNotificationResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- Code – ok

In case of failure or error, see section 1.4.7

FROM RELEASE 5 ON:
The bulk transfer functionality allows an organisation to transfer claims to another organisation of the same type (CR to CR, COMP to COMP).
Every claim is always handled by a CR (i.e. CR1) and a COMP (COMP1). When a claim is transferred from CR1 to CR2, the third party organisation (COMP1) is indirectly involved in the transfer and must be advised that other party is changing. Similarly, when a claim is transferred from COMP1 to COMP2, the third party organisation (CR1) is indirectly involved in the transfer and must be advised that other party is changing.
In order to inform the third party organisations involved in a bulk transfer process, new notifications are generated and sent only to their A2A. From Release 5 on, the getNotificationList request provides the functionality to delete the notification from the system.

FROM GetToken() RELEASE ON:

As the notifications belong to the A2A user the getNotificationList() request must be sent using an ad hoc generated token where User = UserAsID.
In order to avoid receiving an error when calling the getNotificationList() using the token generated with User = UserAsID the parameter "A2Anotification" **must be set to TRUE**. If the parameter is set to FALSE or the parameter is NOT INCLUDED in the request the system will return the following error "Invalid Token. User and UserAsID cannot be the same for this method", see paragraph 1.4.7 for further details.

Please note that this is valid only for the getNotificationsList() and the **removeNotification**() for all other calls the provided token generated with the User = UserAsID will return an error.

2.2.10.3 RemoveNotification

This request provides the functionality to delete the notification from the system.

INPUT:

- accessToken – accessToken for the A2A
- notificationID – the id of the notification to be deleted
- A2ANotification – flag to delete the A2A notifications
-

2.2.10.4 RemoveNotificationResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- Code – ok

In case of failure or error, see section 1.4.7

2.2.11 Reject Claim to CR

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-11		**High**	**0,1,2,3,4,5***

System status before execution:

Can be executed by:	**COMP**
Phase applicable:	**Claim Submitted**

System status after execution:

Phase reached:	**Releases 0,1,2,3,4:** **Claim rejected to CR** **From Release 5 on:** **Acknowledge Rejected Claim**

2.2.11.1 RejectClaimToCR

This request allows a CM to send the claim back to the CR.
From Release 5 on the CM must indicate the reason code for the rejection in the Input.
For all the claims created under the old releases a default reason code (i.e. "0") must be passed in the Input.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
  - rejectionReasonCode (Input field introduced in Release 5)

2.2.11.2 RejectClaimToCRResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- code
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.12 Acknowledge Rejected Claim

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-12	**AcknowledgeRejectedClaim**	**High**	**5,6**

System status before execution:

Can be executed by:	**CR**
Phase applicable:	**Acknowledge Rejected Claim**

System status after execution:

Phase reached:	**Claim rejected to CR**

2.2.12.1 AcknowledgeRejectedClaim

This request provides the functionality to allow a CR to indicate to the portal that they have read the rejection reason and they want to re-send the claim to the compensator.

Example (feasible scenario):

1) The CR adds a claim and sends it to a Compensator
2) The Compensator looks at it and rejects the claim (adding a reason)
3) At this point, the CR (that has received a Notification of "Rejected claim" can also see the claim in the worklist with the phase "Acknowledge Rejected Claim") with the reason for the rejection and they can decide whether to keep the claim in the process in order to re-send it or to exit it from the process.

In this case the CR decides to keep the claim in the process.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- claimXML – the content of the claim notification form (see schema DocumentInput.xsd).

2.2.12.2 AcknowledgeRejectedClaimResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- code
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.13 Exit Rejected Claim

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-13	**ExitRejectedClaim**	**High**	**5,6**

System status before execution:

Can be executed by:	**CR**
Phase applicable:	**Acknowledge Rejected Claim**

System status after execution:

Phase reached:	**Claim Rejected End** *(End of the process)*

2.2.13.1 ExitRejectedClaimRequest

This request allows a CR to exit a rejected claim.

Example (feasible scenario):

1) The CR adds a claim and sends it to a Compensator
2) The Compensator looks at it and rejects the claim (adding a reason)
3) At this point, the CR (that has received a Notification of "Rejected claim" and can also see the claim in the worklist with the phase "Acknowledge Rejected Claim") can see the rejection reason and decides whether to keep the claim in the process in order to re-send it or to exit it from the process.
4) In this case the CR decides to exit the claim from the process.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.2.13.2 ExitRejectedClaimResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- code
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.14 Resend Rejected Claim

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-14	**ResendRejectedClaim**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phase applicable:	**Claim rejected to CR**

System status after execution:

Phase reached:	**Claim Acknowledgement**

2.2.14.1 ResendRejectedClaim

This request provides the functionality for a CR to re-send a claim that has been rejected by a Compensator

Example (feasible scenario):

1) The CR adds a claim and sends it to a Compensator
2) The Compensator looks at it and rejects the claim
3) At this point, the CR (that has received a Notification of "Rejected claim" and can also see the claim in the worklist with the phase "Claim rejected to CR") can modify the CNF (by changing also the ID of the Insurer to which they want to send the claim) and re-send it through this function.

Note 1: if someone interferes with the application via the web portal and moves it in an intermediate tab to edit the claim and resend it via web, this function is no longer usable via A2A because the phase changes from "Claim Rejected to CR" to "Claim Data Input/Edit". Hence the web user must complete the task via web portal as a workaround.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- claimXML – the content of the claim notification form (see schema DocumentInput.xsd).

2.2.14.2 ResendRejectedClaimResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- code
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.15 Reassign to another CM

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-15	**ReassignToAnotherCM**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**COMP**
Phase applicable:	**Claim Acknowledgement** **Claim Submitted**

System status after execution:

Phase reached:	**Claim Acknowledgement** **Claim Submitted**

2.2.15.1 ReassignToAnotherCM

This request allows a CM to re-assign the claim to another CR.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- OrganisationPath – The Path of the compensator to which the claim must be re-assigned

2.2.15.2 ReassignToAnotherCMResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- code
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.16 Acknowledge Claim

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-16	**AcknowledgeClaim**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**COMP**
Phase applicable:	**Claim Acknowledgement**

System status after execution:

Phase reached:	**Claim Submitted**

2.2.16.1 AcknowledgeClaim

This request allows a Compensator to officially acknowledge the receipt of the claim.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.2.16.2 AcknowledgeClaim Response

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- code
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.17 Accept Claim

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-17	**AcceptClaim**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**COMP**
Phase applicable:	**Claim Submitted**

System status after execution:

Phase reached:	**Liability Decision**

2.2.17.1 AcceptClaim

This request allows a Compensator to officially indicate that the claim is theirs. From this moment on, a Compensator cannot re-assign the claim to another compensator and cannot resend it back to the CR.
The Compensator can optionally indicate the Defendant Insurer Claim Reference Number when accepting the claim.

INPUT:

- accessToken – accessToken for the A2A
- insurerClaimReferenceNumber – optional value for the Defendant Insurer Claim Reference Number
- claimData
  - applicationID
  - activityEngineGuid

2.2.17.2 AcceptClaim Response

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- code
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.18 Send Liability Decision

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-18	**SendLiabilityDecision**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**COMP**
Phase applicable:	**Liability decision**

System status after execution:

Phase reached:	One of: **Liability admitted** **Liability not admitted** **Liability admitted with negligence** **Liability admitted (childClaim)**

2.2.18.1 SendLiabilityDecision

This request allows a CM to send the response about the Liability for the given Claim

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- insurerResponseXml – the XML of the response (It must be passed as a string. Please refer to the schema and the WSDL for further info)

2.2.18.2 SendLiabilityDecisionResponse

This is the snychronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- code
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.19 Acknowledge Denied Liability

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-19	**AcknowledgeDeniedLiability**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Liability not admitted**

System status after execution:

Phase reached:	**Stage 1 complete**

2.2.19.1 AcknowledgeDeniedLiability

This command is needed to ensure the CR acknowledges the liability decision taken by the COMP.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.2.19.2 AcknowledgeDeniedLiabilityResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.20 Allocate Claim to Branch

Requirement ID	Requirements Description	Prioritization	Compatibility level
FUN001-20	**AllocateClaimToBranch**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**COMP**
Phases applicable:	*In phases where the Web shows the "AllocateToBranch" button, which are:* ClaimAcknowledgement ClaimSubmitted LiabilityDecision InterimPaymentDecision DecisionForPartialInterimPayment Stage2SettlementPackDecision Stage2SettlementPackCounterOfferByCM Stage2SettlementPackAgreementDecisionComp CourtProceedingsPackFormResponse

System status after execution:

Phase reached:	*Same phase as before*

2.2.20.1 AllocateClaimToBranch

This request allows a CM to allocate the claim to a branch of their organisation.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- branchId

2.2.20.2 AllocateClaimToBranchResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.21 State Fraud

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-21	**StateFraud**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**COMP**
Phases applicable:	*In phases where the Web shows the "StateFraud" button, which are:* ClaimAcknowledgement ClaimSubmitted LiabilityDecision InterimPaymentDecision DecisionForPartialInterimPayment Stage2SettlementPackDecision

System status after execution:

Phase reached:	**Fraud Stated**

2.2.21.1 StateFraud

This request allows a CM to throw the claim out of the process due to a Fraud, adding also a reason for this action.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- reasonCode
- reasonDescription

2.2.21.2 StateFraudResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.22 Acknowledge Fraud Stated

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-22	**AcknowledgeFraudStated**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Fraud Stated**

System status after execution:

Phase reached:	**Fraud Stated End**, *End of the process*

2.2.22.1 AcknowledgeFraudStated

This request allows a CR to simply indicate that they saw the message about the Fraud stated by the Compensator

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.2.22.2 AcknowledgeFraudStatedResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.23 Add attachment

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-23	**AddAttachment**	**High**	**0,1,2,3,4,5,6**

2.2.23.1 AddAttachment

This request provides the functionality to add an attachment to a particular claim. In case of file, the attachment has to be passed to the interface as a stream of bytes; in case of note, the note is passed as text. The attachment is identified in the system with an attachment ID, a title and a description.
The maximum size allowed is 4 MB for each attachment.

The maximum allowed attachments size is set to 20 MB per each single claim. In this way it is possible to upload up to 5 full-size attachments of 4MB each per single claim, or any other combination of attachment size (subject to the 4 MB individual and 20 MB total size limits).

FROM RELEASE 6 ON: a maximum number of attachments has been introduced for Claimant Reps and for Compensators:

- Max Number of attachments allowed of Claimant Reps: 40 attachments
- Max Number of attachments allowed for Compensators: 10 attachments

This means that the Claimant Reps can attach up to 40 attachments to each claim and the Compansators can attach up to 10 attachments to each claims irrespective of the number of attachments added by the counterpart (subject to the 4 MB individual and 20 MB total size limits).

INPUT:

- accessToken – accessToken for the A2A
- Attachment
  - applicationId – the ID of the Claim to which the attachment is related
  - dataAttachmentGuid – *ignore it, optional*
  - dataAttachmentFileName – the filename, empty if attaching a note
  - dataAttachmentFileZip – stream of bytes base64 encoded, representing the file to be uploaded (not compressed with Zip): empty if attaching a note
  - dataAttachmentTitle - notes describing the file content and usage
  - dataAttachmentDesc – a text description of the file

If adding a note, the text will go in dataAttachmentDesc, and leave empty dataAttachmentFileName and dataAttachmentFileZip.

2.2.23.2 addAttachmentResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- code - ok
- value – the ID of the Attachment

In case of failure or error, see section 1.4.7

2.2.24 Get attachments list

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-24	**GetAttachmentsList**	**High**	**0,1,2,3,4,5,6**

2.2.24.1 GetAttachmentsList

This request provides the functionality to get the list of attachments of a particular claim.

INPUT:

- accessToken – accessToken for the A2A
- applicationID **–** the ID of the Claim for which the attachments are requested

2.2.24.2 GetAttachmentsListResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- code
- attachmentsList
  the list of the Attachments for that applicationID
  For each item:
  - dataAttachmentGuid – the Id of the attachment
  - dataAttachmentFileName – the filename of the attachment
  - dataAttachmentTitle - a title for the file
  - dataAttachmentDesc – a description of the file
  - dataAttachmentOwner – the owner of the attachment
  - dataAttachmentDate – the date of the attachment

In case of a note attachment, dataAttachmentFileName and dataAttachmentFileZip will be empty.

In case of failure or error, see section 1.4.7

2.2.25 Get attachment

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-25	**GetAttachment**	**High**	**0,1,2,3,4,5,6**

2.2.25.1 GetAttachment

This request provides the functionality to get an attachment as a stream of bytes.

INPUT:

- accessToken – accessToken for the A2A
- attachmentGuid **–** the ID of the attachment

2.2.25.2 GetAttachmentResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- code – ok
- attachment
  - dataAttachmentGuid – the Id of the attachment
  - dataAttachmentFileName – the filename of the attachment
  - dataAttachmentFileZip – stream of bytes base64 encoded, representing the file content: empty in case of note
  - dataAttachmentTitle - a title for the file
  - dataAttachmentDesc – a description of the file
  - dataAttachmentOwner – the owner of the attachment
  - dataAttachmentDate – the date of the attachment

In case of failure or error, see section 1.4.7

2.2.26 Get printable documents list

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-26	**GetPrintableDocumentsList**	**High**	**0,1,2,3,4,5,6**

2.2.26.1 getPrintableDocumentsList

This request provides the functionality to get the list of printable documents of a particular claim.
Printable documents are added only when the claim changes ownership; as a consequence, when getting ownership of a claim, it's possible to download new printable documents, and while maintaining ownership of the claim there is no need to check for further printable documents list changes.

INPUT:

- accessToken – accessToken for the A2A
- applicationID **–** the ID of the Claim for which the printable documents are requested

2.2.26.2 getPrintableDocumentsListResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- code - ok
- attachmentsList
  the list of the printable documents for that applicationID
  For each item:
    - applicationId – the ID of the Claim to which the attachment is related
    - dataAttachmentGuid – the Id of the Printable document
    - dataAttachmentFileName – the file name of the printable document (including ".PDF" extension)
    - dataAttachmentTitle - a title for the file
    - dataAttachmentDesc – a description of the file

In case of failure or error, see section 1.4.7

2.2.27 Get Printable Document

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-27	**getPrintableDocument**	**High**	**0,1,2,3,4,5,6**

2.2.27.1 getPrintableDocument

This request provides the functionality to get a Printable Document.
Printable documents are added only when the claim changes ownership; as a consequence, when getting ownership of a claim, it's possible to download new printable documents, and while maintaining ownership of the claim there is no need to check for further printable documents list changes.

NOTE: the printable document with title "*Defendant only Claim Notification Form*" can be downloaded only by the CR: COMP are not allowed to download this printable document.

INPUT:

- accessToken – accessToken for the A2A
- PrintableDocumentID **–** the ID of the Printable Document

2.2.27.2 getPrintableDocumentResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- code – ok
- attachment
    - applicationId – the ID of the Claim to which the attachment is related
    - dataAttachmentGuid – the Id of the Printable document
    - dataAttachmentFileName – the file name of the printable document (including ".PDF" extension)
    - dataAttachmentFileZip – stream of bytes base64 encoded, representing the document (this is always a PDF document)
    - dataAttachmentTitle - a title for the file
    - dataAttachmentDesc – a description of the file

In case of failure or error, see section 1.4.7

2.2.28 Lock Claim

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-28	**LockClaim**	**High**	**0,1,2,3,4,5,6**

2.2.28.1 LockClaim

This request provides the functionality to lock a claim.
Only the user who has locked the claim can then execute commands which change the flow step or adds data to the claim. The claim can be unlocked by the same user who has locked using an unlock claim method or can be unlocked by other users using the force unlock method.
The life scope of the lock established by LockClaim is up to the execution of the next command which changes the flow step or adds data to the claim; to maintain the lock after such a command, a new LockClaim should be executed.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.2.28.2 LockClaim Response

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- Code – ok

In case of failure or error, see section 1.4.7

2.2.29 Unlock Claim

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-29	**UnlockClaim**	**High**	**0,1,2,3,4,5,6**

2.2.29.1 UnlockClaim

This request provides the functionality to unlock a claim.
Only the user who locked the claim can unlock it.

INPUT:

- accessToken – accessToken for the A2A
- applicationID

2.2.29.2 UnlockClaim Response

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- Code – ok

In case of failure or error, see section 1.4.7

2.2.30 Force unlock Claim

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-30	**ForceUnlockClaim**	**High**	**0,1,2,3,4,5,6**

2.2.30.1 ForceUnlockClaim

This request provides the functionality to force the unlocking of any claim, currently locked by any user.

INPUT:

- accessToken – accessToken for the A2A
- applicationID

2.2.30.2 ForceUnlockClaimResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code – ok

In case of failure or error, see section 1.4.7

2.2.31 Search Compensators

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-31	**SearchCompensators**	**High**	**0,1,2,3,4,5,6**

2.2.31.1 SearchCompensators

This request provides the functionality to search for a compensator, in order to get the OrganisationId to send the claim. The search is performed with a partial match on the organisationName.

WARNING:
This command is deprecated since the search results do not mimic the behaviour of the web portal: this means that it does not correctly access the mapping between brands and insurers/third party insurers. To ensure the correct routing of the claim to the relevant insurer, this functionality should not be used and the client must use the SearchCompensatorsByInsurerIndex() command, see section 2.2.32 for more details. This command may be removed in the future,

INPUT:

- accessToken – accessToken for the A2A
- compensatorType – I (Insurer), S (Self Insured), M (MIB)
- organisationName

2.2.31.2 SearchCompensatorsResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- Code – ok
- organisationsInfoList. Per each item:
  - organisationId
  - organisationName
  - organisationPath

Please note that the return set is limited in size for performance reasons to a maximum of 50 entries, this function should not be used in an attempt to retrieve and cache locally the whole table of compensators, but only for real-time searches.

In case of failure or error, see section 1.4.7

2.2.32 SearchCompensatorsByInsurerIndex

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-32	**SearchCompensatorsByInsurerIndex**	**High**	**0,1,2,3,4,5,6**

2.2.32.1 SearchCompensatorsByInsurerIndex

This command searches for a compensator using the InsurersIndex table, in order to get the OrganisationId to send the claim to and the Insurer Name to be inserted in the CNF in the field "InsurerName" of the Defendant's Insurer.

Each entry of the Insurer Index table is made of the following fields:
1 and 2) Fields "Insurer name" and "Contact name", used by the system to perform a text search, contains values to help the Claimant Representative finding the right entry
3) field "raPIdInsurerID", which is the ID of the actual Insurer to which the claim must be sent
4) field "raPIdInsurerName", which contains the text to be shown on the CNF

INPUT:

- accessToken – accessToken for the A2A
- OrganisationName

2.2.32.2 SearchCompensatorsByInsurerIndexResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- Code – ok
- InsurerIndexList. Per each item:
  - InsurerName
  - ContactnName
  - rapidInsurerId – the Id to which the claim should be sent, if the user believes that the InsurerName and ContactName are the right ones
  - rapidInsurerPath – the path of the Insurer to which the claim should be sent (note that the path is present only for completion's sake, but it is going to be deprecated, since the organisationID is the important field to add a CNF)
  - rapidInsurerName – the text to be shown on the CNF in the first page of the pdf ("Insurer name").

Please note that the return set is limited in size for performance reasons to a maximum of 50 entries, hence is not intended to be used to retrieve and cache locally the whole InsurerIndex table, but only for real-time searches.

In case of failure or error, see section 1.4.7

2.2.33 ExitProcess

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-33	**ExitProcess**	**High**	**0,1,5,6**

System status before execution:

Can be executed by:	**CR/COMP**
Phases applicable:	*In phases where the Web shows the "ExitProcess" button, which are:* *For CR:* StartOfStage21 InterimSettlementPackForm AcceptanceOfPartialInterimPayment InterimPaymentDecisionTimeout WaitingForInterimPayment Stage2SettlementPackForm Stage2SettlementPackCounterOfferDecision Stage2SettlementPackCounterOffer Stage2SettlementPackAgreementDecision CourtProceedingsPackForm *For COMP:* ClaimAcknowledgement ClaimSubmitted LiabilityDecision InterimPaymentDecision DecisionForPartialInterimPayment Stage2SettlementPackDecision Stage2SettlementPackCounterOfferByCM Stage2SettlementPackAgreementDecisionComp CourtProceedingsPackFormResponse

System status after execution:

Phase reached:	**Exit Process**

2.2.33.1 ExitProcess

This command allows the Compensator or the CR to take a claim out of the process (e.g. if the value is found to be below £1.000 or above the upper limit) when the claim is in their respective worklist. The Reason Code, and the user exiting the claim, the date and the Stage at exit will be retained for MI reporting.
No further actions can be performed on the claim.

INPUT:

- claimData
  - applicationID
  - activityEngineGuid
- accessToken – accessToken for the A2A
- exitReasonCode – the reason for the exit (see Tech Specs of the schema of the GetClaim for the list of reason codes that can be input by the Compensator and by the Claimant Representative):

The Compensator can use these codes: 1,2,3,6,7,9,10

1 - Incomplete information provided on CNF
(please note that this code is valid only during Stage 1)
2 - Value of claim below £1,000
3 - Value of claim exceeds the upper limit
6 - Duplicated claim
7 - Withdrawal of offer
9 - Claim requires further investigation
10 - Other

The Claimant Representative can use these codes: 2,3,4,5,6,7,8,10,11,12,13,14,15

2 - Value of claim below £1,000
3 - Value of claim exceeds the upper limit
4 - Claim is too complex for process
5 - Withdrawal of claim
6 - Duplicated claim
7 - Withdrawal of offer
8 - Interim payment for child claimant
10 – Other
11 – Stage 1 costs not paid on time
(please note that this code is allowed ONLY in the following phases:
- *Stage 2 Settlement Pack confirmation*
- *Stage 2 Settlement Pack counter offer decision*
- *Stage 2 Settlement Pack counter offer*
- *Stage 2 Settlement Pack agreement decision*
- *Stage 2 Settlement Pack agreed*
- *Court Proceedings Pack Form*

ONLY IF Liability admitted – adult claim
AND
ONLY AFTER the time period of 10 business days from the date sent of the Stage 2 Settlement Pack to the Compensator has been reached)

12 – Interim Payment partial offer not accepted
(please note that this code is valid only in the phase AcceptanceOfPartialInterimPayment)
13 – Interim Payment request not answered and/or paid on time
(please note that this code is valid only in the phase InterimPaymentDecisionTimeout*)*
14 - Failure to provide adequate loss of earnings details
15 - Failure to acknowledge CNF on time

- exitComment – free text to add some specific comments

2.2.33.2 ExitProcessResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
In case of success, the system responds with:
- Code - ok
- claimInfo
  - applicationID

- activityEngineGuid
- phaseCacheId
- phaseCacheName

In case of failure or error, see section 1.4.7

Note:
When a not existing Reason Code is used or when a non-valid Reason Code is used (i.e. a CR cannot use the Reason Code 1, the COMP cannot use Reason Code 1 during Stage 2) the system responds with the error message "The reason code is not valid"

2.2.34 AcknowledgeExitProcess

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-34	**AcknowledgeExitProcess**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR/COMP**
Phases applicable:	**Exit Process**

System status after execution:

Phase reached:	**Exit Process End** *(End of the process)*

2.2.34.1 AcknowledgeExitProcess

This request allows a CR/COMP to simply indicate that they saw the message about the Exit Process stated by the other organisation

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.2.34.2 AcknowledgeExitProcessResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.35 Allocate User (executed by CR)

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-35	**AllocateUser**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	*In phases where the Web shows the "AllocateUser" button:* LiabilityAdmitted StartOfStage21 InterimSettlementPackForm AcceptanceOfPartialInterimPayment InterimPaymentDecisionTimeout WaitingForInterimPayment Stage2SettlementPackForm Stage2SettlementPackCounterOfferDecision Stage2SettlementPackCounterOffer Stage2SettlementPackAgreementDecision CourtProceedingsPackForm

System status after execution:

Phase reached:	*The same phase as before*

2.2.35.1 AllocateUser

This request allows a CR to allocate a claim to a specific user, can be executed only if the UserAsID account has the CR Team Leader profile.
This will impact the dispatch of the claim.
To deallocate the claim from a specific user, see DeAllocateUser (section 2.2.37).

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- targetUserID

2.2.35.2 AllocateUserResponse

This is the synchronous response message sent by the system back to the A2A client
OUTPUT
When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.36 Allocate User (executed by COMP)

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-36	**AllocateUser**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**COMP**
Phases applicable:	*In phases where the Web shows the "AllocateUser" button:* ClaimAcknowledgement ClaimSubmitted LiabilityDecision InterimPaymentDecision DecisionForPartialInterimPayment Stage2SettlementPackDecision Stage2SettlementPackCounterOfferByCM Stage2SettlementPackAgreementDecisionComp CourtProceedingsPackFormResponse

System status after execution:

Phase reached:	*The same phase as before*

2.2.36.1 AllocateUser

This request allows a COMP to allocate a claim to a specific user, can be executed only if the UserAsID account has the COMP Claim handler Team Leader profile.
This will impact the dispatch of the claim.
Please note that the target user should have one of the following profiles: "COMP Claim Handler", "COMP Team Leader", "COMP Branch Claim Handler" (but only if they belong to the same branch).
To deallocate the claim from a specific user, see AllocateClaimToBranch (section 0).

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- targetUserID

2.2.36.2 AllocateUserResponse

This is the synchronous response message sent by the system back to the A2A client
OUTPUT
When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.37 DeAllocate User

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-37	**DeAllocateUser**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	*Any phase belonging to the CR*

System status after execution:

Phase reached:	*The same phase as before*

2.2.37.1 DeAllocateUser

This request allows a CR to de-allocate a claim from a specific user; can be executed only if the UserAsID account has CR the Team Leader profile.
This will impact the dispatch of the claim and the notifications related to such claim.
Please note that for COMP, should be used instead AllocateClaimToBranch (see section 0).

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.2.37.2 DeAllocateUserResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.38 Get My Users list

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-38	**GetMyUsersList**	**High**	**0,1,2,3,4,5,6**

2.2.38.1 GetMyUsersList

This request provides the functionality to just retrieve the list of users belonging to the organisation of the calling user.

INPUT:

- accessToken – accessToken for the A2A

2.2.38.2 GetMyUsersListResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- myUsersListResponse
  - Code – ok
  - myUsersList
    the list of users belonging to the organisation (and sub-branches) of the user "userAsID". Per each list item:
    - userId the username
    - userName
    - usrSecondName
    - userSurname
    - userProfile
    - userOrganisationId (of course it is always the same, being the organisation of the userAsID)
    - userBranchId – id of the branch to which the retrieved user belongs
    - userEnabled – flag that indicates whether the user is enabled or disabled

In case of failure or error, see section 1.4.7

2.2.39 AcknowledgeLiabilityDecisionTimeout

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-39	**AcknowledgeLiabilityDecisionTimeout**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Liability Decision Timeout**

System status after execution:

Phase reached:	**Liability Decision Timeout End**, the claims ends

2.2.39.1 AcknowledgeLiabilityDecisionTimeout

This command is needed to ensure the CR acknowledges that the claim reached the time limit to take a liability decision.
As a consequence the claim moves into the phase Liability Decision Timeout End with the Status "END", which removes the claim from the worklist (within the old Releases 0 and 1 the phase remained set to Liability Decision Timeout. The changed behaviour makes it easier for A2A users to filter out claims already acknowledged).

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.2.39.2 AcknowledgeLiabilityDecisionTimeoutResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.40 GetSystemProcessVersion

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-40	**GetSystemProcessVersion**	**High**	**0,1,2,3,4,5,6**

2.2.40.1 GetSystemProcessVersion

This command just returns the most recent process version running in the system (in other words it is the version under which a new claim is created within Claims Portal; of course claims created with older versions of the process are still present in the workflow).

This function helps the Claimant Representative software houses to implement a mechanism that automates the switch from the previous version of the AddClaim, to the new one, in order to deploy their software to their customers in Production before the date of Go Live.

The usage of this command is to be executed before each AddClaim() command: in this way the software will be automatically aware of which AddClaim() version should be executed.
To minimize the impact on the system of calling this command, follow this approach: close to the new release deploy in Production, it will be publicly communicated the target process version that will implement the new release. So the software will start calling this command before each AddClaim() and using the AddClaim() for the legacy release, until the new process version is returned: when this happens, it means the switch has occurred - the new release has been deployed, so the software can stop calling this command and is aware that it should execute the AddClaim() for the new release.

INPUT:

- accessToken – accessToken for the A2A

2.2.40.2 GetSystemProcessVersionResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- Code - ok
- systemProcessVersion
    - processName: this is a fixed string "EL/PL Process"
    - versionMajor: this is an integer
    - versionMinor: this is an integer

In case of failure or error, see section 1.4.7

2.2.41 Acknowledge Liability Admitted With Negligence

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-41	**AcknowledgeLiabilityAdmittedWithNeg**	**High**	**All**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Liability admitted with negligence**

System status after execution:

Phase reached:	**Stage 1 complete**

2.2.41.1 AcknowledgeLiabilityAdmittedWithNeg

This command is needed to ensure the CR acknowledges the liability decision taken by the COMP.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.2.41.2 AcknowledgeLiabilityAdmittedWithNegResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.42 Acknowledge Liability Admitted For Child

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-42	**AcknowledgeLiabilityAdmittedForChild**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Liability admitted (Child claim)**

System status after execution:

Phase reached:	**Start of Stage 2.1**

2.2.42.1 AcknowledgeLiabilityAdmittedForChild

This command is needed to ensure the CR acknowledges the liability decision taken by the COMP.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.2.42.2 AcknowledgeLiabilityAdmittedForChildResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.2.43 Acknowledge Liability Admitted

Requirement ID	Requirements Description	Prioritization	Releases
FUN001-43	**AcknowledgeLiabilityAdmitted**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Liability admitted**

System status after execution:

Phase reached:	**Start of Stage 2.1**

2.2.43.1 AcknowledgeLiabilityAdmitted

This command is needed to ensure the CR acknowledges the liability decision taken by the COMP.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.2.43.2 AcknowledgeLiabilityAdmittedResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.3 Functionalities specific for Stage 2.1

2.3.1 SetInterimPaymentNeeded

Requirement ID	Requirements Description	Prioritization	Releases
FUN002-1	**SetInterimPaymentNeeded**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Start of stage 2.1**

System status after execution:

Phase reached:	*If(isIntermPaymentNeeded = true)* **Interim Settlement Pack Form** *else* **Stage 2 Settlement Pack Form**

2.3.1.1 SetInterimPaymentNeeded

This request allows the functionality to add an Interim Settlement Pack Form Request for a claim.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- isInterimPaymentNeeded – a Boolean flag (true/false) that indicates whether the InterimPayment is needed.

2.3.1.2 SetInterimPaymentNeededResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- claimInfoResponse
  - code – ok
  - claimInfo
    - activityEngineGuid
    - applicationId
    - phaseCacheId
    - phaseCacheName

In case of failure or error, see section 1.4.7

2.3.2 AddInterimSPFRequest

Requirement ID	Requirements Description	Prioritization	Releases
FUN002-2	**AddInterimSPFRequest**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Interim Settlement Pack Form**

System status after execution:

Phase reached:	**Interim Payment decision**

2.3.2.1 AddInterimSPFRequest

This request provides the functionality to add an Interim Settlement Pack Form Request for a claim.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- ISPFRequestXML – the content of the Interim Settlement Pack Form Request.

2.3.2.2 AddInterimSPFRequestResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- claimInfoResponse
  - code – ok
  - claimInfo
    - activityEngineGuid
    - applicationId
    - phaseCacheId
    - phaseCacheName

In case of failure or error, see section 1.4.7

2.3.3 AddInterimSPFResponse

Requirement ID	Requirements Description	Prioritization	Releases
FUN002-3	**AddInterimSPFResponse**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**COMP**
Phases applicable:	**Interim Payment decision**

System status after execution:

Phase reached:	*If(COMP offers the same amount request by CR)* **Waiting for Interim Payment** *else* **Acceptance of partial Interim payment**

2.3.3.1 AddInterimSPFResponse

This request provides the functionality to add an Interim Settlement Pack Form Response for a claim.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- ISPFResponseXML – the content of the Interim Settlement Pack Form Response.

2.3.3.2 AddInterimSPFResponseResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- claimInfoResponse
  - code – ok
  - claimInfo
    - activityEngineGuid
    - applicationId
    - phaseCacheId
    - phaseCacheName

In case of failure or error, see section 1.4.7

2.3.4 Set Stage2_1 Payment

Requirement ID	Requirements Description	Prioritization	Releases
FUN002-4	**SetStage21Payment**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Waiting for Interim payment**

System status after execution:

Phase reached:	*If(InterimPaymentRequested>1000, COMP offered less money, CR did not accepted the partial offer)* **Stage 2.1 complete** *else* **Start of Stage 2.1**

2.3.4.1 SetStage2_1Payment

This request allows a CR to simply indicate that the Payment for the given Interim Settlement Pack Form has been received or not.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- isStage2_1Paid – 0/1 (or 'false'/'true) boolean
  NOTE that "isStage2_1Paid = false" can be accepted by the system only if the period of "10 (or 15)" business days is expired.

2.3.4.2 SetStage2_1PaymentResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.3.5 AcceptPartialInterimPayment

Requirement ID	Requirements Description	Prioritization	Releases
FUN002-5	**AcceptPartialInterimPayment**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Acceptance of partial Interim payment**

System status after execution:

Phase reached:	**Decision for Partial Interim payment**

2.3.5.1 AcceptPartialInterimPayment

This request allows a CR to accept or not to accept the offer for a partial payment made by the Compensator.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- isPartialInterimPaymentAccepted – 0/1 (or 'false'/'true') boolean

2.3.5.2 AcceptPartialInterimPaymentResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.3.6 AcknowledgePartialPaymentDecision

Requirement ID	Requirements Description	Prioritization	Releases
FUN002-6	**AcknowledgePartialPaymentDecision**	**High**	**0,1,2,3,4, 5,6**

System status before execution:

Can be executed by:	**COMP**
Phases applicable:	**Decision for partial Interim payment**

System status after execution:

Phase reached:	**Waiting for Interim payment**

2.3.6.1 AcknowledgePartialPaymentDecision

This request allows a Compensator to simply indicate that they received the message with the decision on the Partial Payment taken by the CR. Nothing else: it is simply a step needed in the workflow.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.3.6.2 AcknowledgePartialPaymentDecisionResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.3.7 AcknowledgeInterimPaymentDecisionTimeout

Requirement ID	Requirements Description	Prioritization	Releases
FUN002-7	**AcknowledgeInterimPaymentDecisionTimeout**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Interim Payment Decision Timeout**

System status after execution:

Phase reached:	**Stage 2 Settlement Pack Form**

2.3.7.1 AcknowledgeInterimPaymentDecisionTimeout

This command is needed to ensure the CR acknowledges that the claim reached the time limit to take an interim payment decision. The claim proceeds to Stage 2 Settlement Pack Form.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.3.7.2 AcknowledgeInterimPaymentDecisionTimeoutResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.3.8 ReturnToStartOfStage21

Requirement ID	Requirements Description	Prioritization	Releases
FUN002-8	**ReturnToStartOfStage21**	**High**	**3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Interim Payment Decision Timeout**

System status after execution:

Phase reached:	**Start of stage 2.1**

2.3.8.1 ReturnToStartOfStage21

This command allows the CR to return to the Start of stage 2.1 in order to ask for an additional Interim payment. The claim proceeds to Start of stage 2.1.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.3.8.2 ReturnToStartOfStage21Response

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.3.9 ExtendInterimPaymentDecisionTimeout

Requirement ID	Requirements Description	Prioritization	Releases
FUN002-9	**ExtendInterimPaymentDecisionTimeout**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**COMP**
Phases applicable:	**Interim Payment Decision** *In case of first Interim payment:* *only if (InterimPaymentRequested>1000)* *In case of SUBSEQUENT Interim payment:* *for any value of InterimPaymentRequested*

System status after execution:

Phase reached:	**Interim Payment Decision**

2.3.9.1 ExtendInterimPaymentDecisionTimeout

This request allows a Compensator to extend the timeframe needed to take a decision for the Interim Settlement Pack Request in the following cases:

- First Interim payment - only if the Interim payment requested by the Claimant representative is greater than £1000.
- Subsequent Interim payments – for any Interim payment value requested by the Claimant representative.

When successful, the date of timeout to take a decision is re-set to 30 business days from the date sent of Interim payment request.

NOTE: it is possible to extend the timeout only once: after the timeout has been successfully extended to 30 business days, it will not be possible to extend it again.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.3.9.2 ExtendInterimPaymentDecisionTimeoutResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.3.10 RejectInterimSettlementPack

Requirement ID	Requirements Description	Prioritization	Releases
FUN002-10	**RejectInterimSettlementPack**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**COMP**
Phases applicable:	**Interim Payment decision**

System status after execution:

Phase reached:	**Interim Settlement Pack rejected**

2.3.10.1 RejectInterimSettlementPack

This request allows the Compensator to reject the request for an Interim payment made by the Claimant Representative.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- rejectionComment – a free text to add a short comment

2.3.10.2 RejectInterimSettlementPackResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.3.11 AcknowledgeRejectedInterimSettlementPack

Requirement ID	Requirements Description	Prioritization	Releases
FUN002-11	**AcknowledgeRejectedInterimSettlement Pack**	**High**	**0,1,2,3,4, 5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Interim Settlement Pack rejected**

System status after execution:

Phase reached:	**Start of Stage 2.1**

2.3.11.1 AcknowledgeRejectedInterimSettlementPack

This request allows the Claimant Representative to simply indicate that they received the message with the rejection of Interim Settlement Pack Form. Nothing else: it is simply a step needed in the workflow.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.3.11.2 AcknowledgeRejectedInterimSettlementPackResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.4 Functionalities specific for Stage 2.2

2.4.1 AddStage2SPFRequest

Requirement ID	Requirements Description	Prioritization	Releases
FUN003-1	**AddStage2SPFRequest**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Stage 2 Settlement Pack Form**

System status after execution:

Phase reached:	**Stage 2 Settlement Pack decision**

2.4.1.1 AddStage2SPFRequest

This request provides the functionality to add a Stage 2 Settlement Pack Form Request for a claim.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- S2SPFRequestXML – the content of the Stage 2 Settlement Pack Form Request.

2.4.1.2 AddStage2SPFRequestResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- claimInfoResponse
  - code – ok
  - claimInfo
    - activityEngineGuid
    - applicationId
    - phaseCacheId
    - phaseCacheName

In case of failure or error, see section 1.4.7

2.4.2 AddStage2SPFResponse

Requirement ID	Requirements Description	Prioritization	Releases
FUN003-2	**AddStage2SPFResponse**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**COMP**
Phases applicable:	**Stage 2 Settlement Pack decision**

System status after execution:

Phase reached:	*If(SettlementPackDecision = "C")* **Stage 2 Settlement Pack confirmation** *If(SettlementPackDecision = "CO")* **Stage 2 Settlement Pack counter offer decision** *If(SettlementPackDecision = "R")* **Stage 2 Settlement Pack repudiation**

2.4.2.1 AddStage2SPFResponse

This request provides the functionality to add a Stage 2 Settlement Pack Form Response for a claim.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- S2SPFResponseXML – the content of the Stage 2 Settlement Pack Form Response.

2.4.2.2 AddStage2SPFResponseResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- claimInfoResponse
  - code – ok
  - claimInfo
    - activityEngineGuid
    - applicationId
    - phaseCacheId
    - phaseCacheName

In case of failure or error, see section 1.4.7

2.4.3 AcknowledgeStage2SPFRepudiation

Requirement ID	Requirements Description	Prioritization	Releases
FUN003-3	**AcknowledgeStage2SPFRepudiation**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Stage 2 Settlement Pack repudiation**

System status after execution:

Phase reached:	**Stage 2 Settlement Pack repudiation End,** *the claim ends*

2.4.3.1 AcknowledgeStage2SPFRepudiation

This request allows a Claimant Representative to simply indicate that they received the message with the decision on the Stage 2 Settlement Pack Form taken by the Compensator. Nothing else: it is simply a step needed in the workflow.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.4.3.2 AcknowledgeStage2SPFRepudiationResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.4.4 AcknowledgeStage2SPFConfirmation

Requirement ID	Requirements Description	Prioritization	Releases
FUN003-4	**AcknowledgeStage2SPFConfirmation**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Stage 2 Settlement Pack confirmation**

System status after execution:

Phase reached:	***IF*** *the claimant was a child on the date of agreement:* **Court Proceedings Pack** ***Else:*** **Stage 2 Settlement Pack confirmation End**, *the claim ends*

2.4.4.1 AcknowledgeStage2SPFConfirmation

This request allows a Claimant Representative to simply indicate that they received the message with the decision on the Stage 2 Settlement Pack Form taken by the Compensator. Nothing else: it is simply a step needed in the workflow.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.4.4.2 AcknowledgeStage2SPFConfirmationResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.4.5 AddStage2SPFCounterOfferByCM

Requirement ID	Requirements Description	Prioritization	Releases
FUN003-5	**AddStage2SPFCounterOfferByCM**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**COMP**
Phases applicable:	**Stage 2 Settlement Pack counter offer (Compensator)**

System status after execution:

Phase reached:	*If(SettlementPackCounterOfferDecision = "NEWCO")* **Stage 2 Settlement Pack counter offer decision** *If(SettlementPackCounterOfferDecision = "ACO")* **Stage 2 Settlement Pack agreed (CR)**

2.4.5.1 AddStage2SPFCounterOfferByCM

This request allows a Compensator to send a Stage 2 Settlement Pack Form Counter Offer for a claim.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- S2SPFCounterOfferByCMXML – the content of the Stage 2 Settlement Pack Form Counter Offer that can be submitted by the Compensator

2.4.5.2 AddStage2SPFCounterOfferByCMResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- claimInfoResponse
  - code – ok
  - claimInfo
    - activityEngineGuid
    - applicationId
    - phaseCacheId
    - phaseCacheName

In case of failure or error, see section 1.4.7

2.4.6 AddStage2SPFCounterOfferByCR

Requirement ID	Requirements Description	Prioritization	Releases
FUN003-6	**AddStage2SPFCounterOfferByCR**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Stage 2 Settlement Pack counter offer**

System status after execution:

Phase reached:	**Stage 2 Settlement Pack counter offer (Compensator)**

2.4.6.1 AddStage2SPFCounterOfferByCR

This request allows a Claimant Representative to send a Stage 2 Settlement Pack Form Counter Offer for a claim.
The reason why there are 2 similar functionalities to add a counter offer is because the set of fields inserted as a counter offer by the CRs is different from the one inserted by the Compensators.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- S2SPFCounterOfferByCRXML – the content of the Stage 2 Settlement Pack Form Counter Offer that can be submitted by the Claimant Representative

2.4.6.2 AddStage2SPFCounterOfferByCRResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- claimInfoResponse
  - code – ok
  - claimInfo
    - activityEngineGuid
    - applicationId
    - phaseCacheId
    - phaseCacheName

In case of failure or error, see section 1.4.7

2.4.7 SetStage2SPFCounterOfferNeeded

Requirement ID	Requirements Description	Prioritization	Releases
FUN003-7	**SetStage2SPFCounterOfferNeeded**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Stage 2 Settlement Pack counter offer decision**

System status after execution:

Phase reached:	*If(isStage2SPFCounterOfferNeeded = true)* **Stage 2 Settlement Pack counter offer** *else* **Stage 2 Settlement Pack agreement decision**

2.4.7.1 SetStage2SPFCounterOfferNeeded

This request allows a Claimant Representative to indicate that they don't need to send to the Compensator a new counter offer for a given claim. As a consequence the claim is moved ahead in the workflow.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- isStage2SPFCounterOfferNeeded – a Boolean flag (true/false) that indicates whether the counter offer is needed or not.

2.4.7.2 SetStage2SPFCounterOfferNeededResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- claimInfoResponse
  - code – ok
  - claimInfo
    - activityEngineGuid
    - applicationId
    - phaseCacheId
    - phaseCacheName

In case of failure or error, see section 1.4.7

2.4.8 ExtendStage2SPFDecisionTimeout

Requirement ID	Requirements Description	Prioritization	Releases
FUN003-8	**ExtendStage2SPFDecisionTimeout**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**COMP**
Phases applicable:	**Stage 2 Settlement Pack decision**

System status after execution:

Phase reached:	**Stage 2 Settlement Pack decision**

2.4.8.1 ExtendStage2SPFDecisionTimeout

This request allows a Compensator to extend the timeframe needed to take a decision for the Staqe2SP Request.
When successful; the date of timeout to take a decision is re-set to a new value, and as a consequence also the Date of Timeout for the counter-offer is recalculated (this is only needed in case there will be a counter offer).

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- newTimeOut – the new date of timeout to be set. It must be set to a date later than the current date of timeout for the S2SP decision (use format of xs:Date even if for technical reasons the WSDL contains xs:DateTime)
- reasonForExtension – the optional integer code of the reason for extension:
  1=No valid CRU certificate
  2=Other agreed extension (if code not specified, this is used as default)
  while this parameter is optional, we strongly suggest to specify it explicitly.

2.4.8.2 ExtendStage2SPFDecisionTimeoutResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- claimInfoResponse
  - code – ok
  - claimInfo
    - activityEngineGuid
    - applicationId
    - phaseCacheId
    - phaseCacheName

In case of failure or error, see section 1.4.7

2.4.9 ExtendStage2SPFCounterOfferTimeout

Requirement ID	Requirements Description	Prioritization	Releases
FUN003-9	**ExtendStage2SPFCounterOfferTimeout**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR/COMP**
Phases applicable:	*CR* → **Stage 2 Settlement Pack counter offer** *CR* → **Stage 2 Settlement Pack counter offer decision** *COMP* → **Stage 2 Settlement Pack counter offer (Compensator)**

System status after execution:

Phase reached:	*the same phase as before*

2.4.9.1 ExtendStage2SPFCounterOfferTimeout

This request allows a Compensator or a CR to extend the timeframe needed to make a counter offer for the Stage2SP.
When successful; the date of timeout to send a counter offer is re-set to a new value.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- newTimeOut – the new date of timeout to be set. It must be set to a date later than the current date of timeout for the counter offer (use format of xs:Date even if for technical reasons the WSDL contains xs:DateTime)

reasonForExtension – the optional integer code of the reason for extension:
1=No valid CRU certificate
2=Other agreed extension (if code not specified, this is used as default)
while this parameter is optional, we strongly suggest to specify it explicitly.

2.4.9.2 ExtendStage2SPFCounterOfferTimeoutResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT
When successful, the system responds with:

- claimInfoResponse
  - code – ok
  - claimInfo
    - activityEngineGuid
    - applicationId
    - phaseCacheId
    - phaseCacheName

In case of failure or error, see section 1.4.7

2.4.10 SetStage2SPFAgreementDecision

Requirement ID	Requirements Description	Prioritization	Releases
FUN003-10	**SetStage2SPFAgreementDecision**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR/COMP**
Phases applicable:	*CR* → **Stage 2 Settlement Pack agreement decision** *COMP* → **Stage 2 Settlement Pack agreement decision (Compensator)**

System status after execution:

Phase reached:	*If the CR sets the agreement decision:* *If (isAgreed = true)* **Stage 2 Settlement Pack agreed** *else* **Court Proceedings Pack** Else if the Compensator sets the agreement decision: *If (isAgreed = true)* **Stage 2 Settlement Pack agreed (CR)** *else* **Stage 2 Settlement Pack not agreed (CR)**

2.4.10.1 SetStage2SPFAgreementDecision

This request allows the Claimant Representative or the Compensator to indicate whether they agree with the S2SPF counter offer or not. As a consequence the claim is moved ahead in the workflow.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- isAgreed – a Boolean flag (true/false) that indicates whether the offer is agreed or not.

2.4.10.2 SetStage2SPFAgreementDecisionResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- claimInfoResponse
  - code – ok
  - claimInfo
    - activityEngineGuid
    - applicationId
    - phaseCacheId
    - phaseCacheName

In case of failure or error, see section 1.4.7

2.4.11 AcknowledgeStage2SPFAgreed

Requirement ID	Requirements Description	Prioritization	Releases
FUN003-11	**AcknowledgeStage2SPFAgreed**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**COMP/CR**
Phases applicable:	*COMP* → **Stage 2 Settlement Pack agreed** *CR* → **Stage 2 Settlement Pack agreed (CR)**

System status after execution:

Phase reached:	***IF*** *the claimant was a child on the date of agreement:* **Court Proceedings Pack** ***Else:*** **Stage 2 Settlement Pack agreed End**, *the claim ends*

2.4.11.1 AcknowledgeStage2SPFAgreed

This request allows the Compensator or the Claimant Representative to simply indicate that they received the message that informed them that the Stage 2 Settlement Pack Form was agreed by the Claimant Representative or the Compensator respectively. Nothing else: it is simply a step needed in the workflow.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.4.11.2 AcknowledgeStage2SPFAgreedResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.4.12 AcknowledgeStage2SPFNotAgreed

Requirement ID	Requirements Description	Prioritization	Releases
FUN003-12	**AcknowledgeStage2SPFNotAgreed**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Stage 2 Settlement Pack not agreed (CR)**

System status after execution:

Phase reached:	**Court Proceedings Pack**

2.4.12.1 AcknowledgeStage2SPFNotAgreed

This request allows the Claimant Representative to simply indicate that they received the message that informed them that the Stage 2 Settlement Pack Form was NOT agreed by the Compensator. Nothing else: it is simply a step needed in the workflow.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.4.12.2 AcknowledgeStage2SPFNotAgreedResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.4.13 AddCPPFRequest

Requirement ID	Requirements Description	Prioritization	Releases
FUN003-13	**AddCPPFRequest**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Court Proceedings Pack Form**

System status after execution:

Phase reached:	**Court Proceedings Pack Form Response**

2.4.13.1 AddCPPFRequestRequest

This request provides the functionality to add a Court Proceedings Pack Form Request for a claim.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- CPPFRequestXML – the content of the Court Proceedings Pack Form Request.

2.4.13.2 AddCPPFRequestResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- claimInfoResponse
  - code – ok
  - claimInfo
    - activityEngineGuid
    - applicationId
    - phaseCacheId
    - phaseCacheName

In case of failure or error, see section 1.4.7

2.4.14 AddCPPFResponse

Requirement ID	Requirements Description	Prioritization	Releases
FUN003-14	**AddCPPFResponse**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**COMP**
Phases applicable:	**Court Proceedings Pack Form Response**

System status after execution:

Phase reached:	**End of Stage 2**

2.4.14.1 AddCPPFResponse

This request provides the functionality to add a Court Proceedings Pack Form Response for a claim.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid
- CPPFResponseXML – the content of the Court Proceedings Pack Form Response.

2.4.14.2 AddCPPFResponseResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- claimInfoResponse
  - code – ok
  - claimInfo
    - activityEngineGuid
    - applicationId
    - phaseCacheId
    - phaseCacheName

In case of failure or error, see section 1.4.7

2.4.15 AcknowledgeCPPFResponse

Requirement ID	Requirements Description	Prioritization	Releases
FUN003-15	**AcknowledgeCPPFResponse**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**End of Stage 2**

System status after execution:

Phase reached:	**End of Stage 2 End,** *the claim ends*

2.4.15.1 AcknowledgeCPPFResponse

This request allows a Claimant Representative to simply indicate that they received the response to the CPPF request. Nothing else: it is simply a step needed in the workflow.

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.4.15.2 AcknowledgeCPPFResponseResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7

2.4.16 AcknowledgeStage2SPFDecisionTimeout

Requirement ID	Requirements Description	Prioritization	Releases
FUN003-16	**AcknowledgeLiabilityDecisionTimeout**	**High**	**0,1,2,3,4,5,6**

System status before execution:

Can be executed by:	**CR**
Phases applicable:	**Stage 2 Settlement Pack Decision Timeout**

System status after execution:

Phase reached:	**Stage 2 Settlement Pack Decision Timeout End,** the claim ends

2.4.16.1 AcknowledgeStage2SPFDecisionTimeout

This command is needed to ensure the CR acknowledges that the claim reached the time limit to take a Stage2 Settlement Pack decision.
As a consequence the claim moves into the phase Stage 2 Settlement Pack Decision Timeout End and enters in the Process Status "END", which removes the claim from the worklist.
(within the old Releases 0 and 1 the phase remained set to Stage 2 Settlement Pack Decision Timeout. The new behaviour makes it easier for A2A users to filter out claims already acknowledged).

INPUT:

- accessToken – accessToken for the A2A
- claimData
  - applicationID
  - activityEngineGuid

2.4.16.2 AcknowledgeStage2SPFDecisionTimeoutResponse

This is the synchronous response message sent by the system back to the A2A client.

OUTPUT

When successful, the system responds with:

- Code - ok
- claimInfo
  - applicationID
  - activityEngineGuid
  - phaseCacheId
  - phaseCacheName

In case of failure or error, see section 1.4.7