CRIF

Claims Portal

INTEGRATION A2A TEST DOCUMENT
CR54 – A2A Token and Change Password - ELPL

Version 1.0

Revision History

Date	Version	Description
26/10/2022	1.0	First version issued

Table of Contents

INTEGRATION TEST CR054/A2A Token

1. Introduction

1.1 Purpose

The aim of this document is to provide to the users that are integrating their system with the Claims Portal system a useful guideline to test the changes introduced with CR54:

Requirement Description
GetToken() functionality
RefreshToken() functionality
New user authentication mechanism for the ELPL methods
ChangePassword() functionality

1.2 Out of scope of this test

These tests are only aimed at testing the changes related to the Authentication mechanism and the new functionalities introduced with CR54, no changes to the flow have been introduced and for this reason test of the workflow are out of scope.
The flow described at paragraph 6 is aimed to test the new login mechanism using the accessToken instead of the old credentials.

1.3 Access and UserIDs

Test environment web site to connect to: https://piptesta2a.crif.com/
Until A2A Token is deployed in production there will be 2 different endpoints available in integration, one where the authentication mechanism has not been changed and one with the new functionalities and the modifications introduced by A2A Token.

OLD UAT Environment	NEW UAT Environment
https://piptesta2a.crif.com/ELPL.WS/ELPLWS	
	https://piptesta2a.crif.com/ELPL.WSTK/ELPLWSTK

1.4 Time frame for testing

The changes on integration environment will be available from w/c 07th November 2022.

1.5 Media used for the test counterpart: A2A or WEB

The steps meant to exercise specific A2A commands are worded with the pattern "Execute the command"; other steps are worded in a generic way: this is because the steps could be done either by A2A commands or by logging on the web portal and it's not relevant for the test case.

In general, if the software is for a CR client, there is no requirement for the tester to implement A2A commands for a COMP to complete each test case, acting on the web portal for COMP steps may be a light and quick solution. On the other side, if the software is for a COMP client, it's not required for the tester to implement A2A commands for the CR, apart from implementing the addClaim() to quickly create a claim instead of filling-in seven tabs of data on the WEB-UI. Provided UserIDs will allow both A2A and Web access.

1.6 ClientID

The clientID is an optional field present in the getToken() and refreshToken() newly introduced functionalities. This field must be used only by the users with a ClientID allocated by Claims Portal, if the ClientID is not available the field must be omitted when invoking the getToken() and refreshToken() and tests 2.7 and 3.8 can be skipped.

2. Test Plan GetToken()

Nr.	Description	Who?	Expected results	Checks	Notes
2.1	Execute a getToken() using the correct userID, password, userAsID, clientID	CR/COMP	The system returns a valid response containing the accessToken, refreshToken, the accessTokenExpiresIn and the refreshTokenExpiresIn	Check that the accessTokenExpiresIn is equal to 7200s and the refreshTokenExpiresIn is equal to 86400s.	
2.2	Execute a getToken() using the same value used at point 2.1, the operation must be performed before two hours have passed since the getToken() of point 2.1	CR/COMP	The system returns a valid response containing the same accessToken and refreshToken returned at point 2.1. The returned values of the accessTokenExpiresIn and the refreshTokenExpiresIn will be decreased equal to the time passed since the getToken() performed at point 2.1	Check that the accessTokenExpiresIn is equal to 7200s – the time passed from the getToken() performed at point 2.1 and the refreshTokenExpiresIn is equal to 86400s – the time passed from the getToken() performed at point 2.1. Check that the accessToken and refreshToken values are the same values returned at point 2.1	**The operation must be performed before two hours have passed since the getToken() performed at point 2.1.**
2.3	Execute a getToken() using the correct userID, password, ClientID and a different userAsID from the one used at point 2.1 and 2.2	CR/COMP	The system returns a valid response containing the accessToken, refreshToken, the accessTokenExpiresIn and the refreshTokenExpiresIn	Check that the accessToken and refreshToken are different from the ones obtained at point 2.1. Check also that the counters returned in the accessTokenExpiresIn and refreshTokenExpiresIn fields are 7200s and 86400s.	
2.4	Execute a getToken() using an invalid userID but valid password,	CR/COMP	The system returns a login failure massage.	Check that the system returns a login failure message	

	userAsID, clientID				
2.5	Execute a getToken() using an invalid password but valid userID, userAsID, clientID	CR/COMP	The system returns a login failure massage.	Check that the system returns a login failure message	
2.6	Execute a getToken() using an invalid userAsID but valid userID, password, clientID	CR/COMP	The system returns a login failure massage.	Check that the system returns a login failure message	**The userAsID must be a non existing username**
2.7	Execute a getToken() using an invalid clientID but valid userID, password, userAsID	CR/COMP	The system returns a login failure massage.	Check that the system returns a login failure message	**Enter a ClientID NOT in the format SHnnnn (SH + four numbers).** **In case of no ClientID assigned by Claims Portal this step can be skipped.**
2.8	Execute a getToken() using the correct userID, password, userAsID, ClientID after 24hours the getToken() performed at point 2.1	CR/COMP	The system returns a valid response containing the accessToken, refreshToken, the accessTokenExpiresIn and the refreshTokenExpiresIn	Check that the accessTokenExpiresIn is equal to 7200s and the refreshTokenExpiresIn is equal to 86400s. Check that the accessToken and the refreshToken values are different from the ones returned by the getToken() at point 2.1	

3.   Test Plan refreshToken()

Nr.	Description	Who?	Expected results	Checks	Notes
3.1	Execute a refreshToken() using the correct username, userAsID, clientID and refreshToken obtained with the getToken() performed at point 2.1	CR/COMP	The system returns a valid response containing the accessToken, refreshToken, the accessTokenExpiresIn and the refreshTokenExpiresIn	Check that the accessTokenExpiresIn is equal to 7200s and the refreshTokenExpiresIn is equal to 86400s – the seconds passed from the getToken() performed at point 2.1. Checks that the accessToken and the refreshToken values are different from the values obtained from the getToken() performed at point 2.1.	
3.2	Execute a refreshToken() using the same userAsID and clientID used at point 3.1, the refreshToken to be used is the refreshToken obtained performing the refreshToken() at point 3.1	CR/COMP	The system returns a valid response containing a new accessToken and a new refreshToken. The returned value of the accessTokenExpiresIn is 7200s and the returned value in the refreshTokenExpiresIn will be decreased equal to the time passed since the refreshToken() performed at point 3.1	Check that the accessTokenExpiresIn is equal to 7200s and the refreshTokenExpiresIn is equal to 86400s – the seconds passed from the refreshToken() performed at point 3.1. Check that the accessToken and the refreshToken values are different from the values obtained from the refreshToken() performed at point 3.1.	**The operation must be performed before two hours have passed since the refreshToken() performed at point 3.1.**
3.3	Execute a refreshToken() using the username, userAsID and clientID used for the getToken() at point	CR/COMP	The system returns a valid response containing the accessToken, refreshToken, the accessTokenExpiresIn and the refreshTokenExpiresIn	Check that the accessTokenExpiresIn is equal to 7200s and the refreshTokenExpiresIn is equal to 86400s – the seconds passed from the getToken()	

				performed at point 2.3. Check that the accessToken and the refreshToken values are different from the values obtained from the getToken() performed at point 2.3 but also different from the values obtained at point 3.1.	
	2.3 and the refreshToken obtained with the getToken() performed at point 2.3				
3.4	Execute a refreshToken using the same credentials used at point 3.3 but with the refreshToken obtained at point 3.1	CR/COMP	The system returns a bad token error message.	Check that the system returns a bad token error message.	
3.5	Execute a refreshToken using the same credentials used at point 3.1 but with the accessToken obtained at point 3.1	CR/COMP	The system returns a bad token error message.	Check that the system returns a bad token error message.	
3.6	Execute a refreshToken() using an invalid userID but valid, userAsID, ClientID and refreshToken	CR/COMP	The system returns a bad token error message.	Check that the system returns a bad token error message.	
3.7	Execute a refreshToken() using an invalid userAsID but valid, userID, ClientID and refreshToken	CR/COMP	The system returns a bad token error message.	Check that the system returns a bad token error message.	**The userAsID must be a non-existing username**

3.8	Execute a refreshToken() using an invalid ClientID but valid, userID, userAsID and refreshToken	CR/COMP	The system returns a login failure massage.	Check that the system returns a login failure message	**Enter a ClientID NOT in the format SHnnnn (SH + four numbers).** **In case of no ClientID assigned by Claims Portal this step can be skipped.**
3.9	Execute a refreshToken() using an invalid refreshToken() but valid, userID, userAsID and clientID	CR/COMP	The system returns a bad token error message.	Check that the system returns a bad token error message.	

4.   Test Plan getToken() and refreshToken() mixed tests

Nr.	Description	Who?	Expected results	Checks	Notes
4.1	Execute a getToken() using the same credentials used at point 2.1.	CR/COMP	The system returns a valid response containing the accessToken, refreshToken, the accessTokenExpiresIn and the refreshTokenExpiresIn		
4.2	After 24h with no other getToken() or refreshtoken() execute a refreshToken() using	CR/COMP	The system returns a bad token error message.	Check that the system returns a bad token error message.	

	the refreshToken obtained by the getToken() performed at point 4.1				
4.3	Execute a getToken() using the same credentials used at point 2.1.	CR/COMP	The system returns a valid response containing the accessToken, refreshToken, the accessTokenExpiresIn and the refreshTokenExpiresIn		
4.4	After 1h45min execute a refreshToken() using the refreshToken obtained with the getToken() performed at point 4.3	CR/COMP	The system returns a valid response containing the accessToken, refreshToken, the accessTokenExpiresIn and the refreshTokenExpiresIn		
4.5	After more than 24 hours from the getToken() performed at point 4.3 execute a refreshToken() using the refreshToken obtained at point 4.4	CR/COMP	The system returns a bad token error message.	Check that the system returns a bad token error message.	
4.6	After the failed refreshToken() performed at point 4.5 execute a getToken() with the same credentials used at point 2.1	CR/COMP	The system returns a valid response containing the accessToken, refreshToken, the accessTokenExpiresIn and the refreshTokenExpiresIn	Check that the accessTokenExpiresIn is equal to 7200s and the refreshTokenExpiresIn is equal to 86400s.	

5.   Test Plan changePassword()

Nr.	Description	Who?	Expected results	Checks	Notes
5.1	Execute a changePassword() using as userID the username of the a2a user, as oldPassword the current password of the a2a user and as newPassword the new password that needs to be set up for the a2a user. The old password of the a2a user must not be expired.	CR/COMP	The system returns a valid response containing an acknowledgement of the change password successfully performed	Check that the system returns a response like: <code>Ok</code> <value>Success</value>	
5.2	Execute a getToken() using the old password used at point 5.1	CR/COMP	The system returns a login failure massage.	Check that the system returns a login failure message	
5.3	Execute a getToken() using the new password set at point 5.1	CR/COMP	The system returns a valid getToken() response.	Check that the system returns a valid getToken() response	
5.4	Execute a changePassword() using a wrong userID, a valid oldPassword and a valid newPassword	CR/COMP	The system returns a login failure massage.	Check that the system returns a login failure message	
5.5	Execute a changePassword() using a valid userID, a wrong oldPassword and a valid newPassword	CR/COMP	The system returns a login failure massage.	Check that the system returns a login failure message	**The oldPassword value must be a value never used as password for the a2a user.**
5.6	Execute a changePassword() using a valid userID, a wrong oldPassword and a valid newPassword	CR/COMP	The system returns a login failure massage.	Check that the system returns a login failure message	**The oldPassword must be a value previously used for the**

					a2a user but not the current value of the password that needs to be changed.
5.7	Execute a changePassword() using as userID the username of the a2a user, as oldPassword the current password of the a2a user and as newPassword the new password that needs to be set up for the a2a user. The a2a user must have an already expired password.	CR/COMP	The system returns a valid response containing an acknowledgement of the change password successfully performed	Check that the system returns a response like: <code>Ok</code> <value>Success</value>	**In order to test this point you will need to contact us, we will force the expiration date of the a2a user password in order to allow you to test this point. Please see below instructions on how to make this request.**

5.1 **Step 5.7 email**

In order to provide you with assistance to test step 5.7 complete the form "A2A Integration/Testing Contact Us - A2A users / Software Houses" available on the Claims Portal web site https://www.claimsportal.org.uk/contact-us/ attach to an email and send to the Help Desk, email address below. Please provide all information requested, if information is missing it may create a delay or result in us being unable to assist you.

SEND TO: helpdesk@rapidclaimsettlement.org.uk
SUBJECT: ELPL A2A Integration Testing Change password expiry date

6.   Test Plan – ELPL – Complete workflow

The getToken() and refreshToken() commands must be called only if the step related to the CR/COMP is performed via A2A, i.e. if the software is a CR client that decided to act as a COMP using the web interface it is not necessary for the tester to implement A2A commands to obtain the tokens for COMP. On the other side, if the software is a COMP client that decided to act as a CR using the web interface, it's not required for the tester to implement A2A commands to obtain the tokens for the CR, unless the COMP decides to implement the addClaim() to quickly create a claim instead of filling-in seven tabs of data on the WEB-UI. In this case the getToken() and refreshToken() commands must be implemented to be able to call the addClaim().

All the steps below need to be completed within 24 hours from the getToken() performed at point 6.1 and/or point 6.2, if these steps are not completed within 24 hours a new getToken() must be performed (both for CR and COMP if both have been implemented). The tokens to be used to complete the following steps (until the next step containing a refreshToken()) are the tokens obtained with the newly performed getToken().

Nr.	Description	Who?	Expected results	Checks	Notes
6.1	Obtain an accessToken and a refreshToken by executing the command getToken()	CR	The system returns an accessToken and a refreshToken for the claimant representative		
6.2	Obtain an accessToken and a refreshToken by executing the command getToken()	COMP	The system returns an accessToken and a refreshToken for the compensator		
6.3	Obtain the current version of the process by executing the command getSystemProcessVersion(). In case of CR use the accessToken obtained at point 6.1, in case of COMP use the accessToken obtained at point 6.2.	CR/COMP			
6.4	Obtain the compensator's details executing the command searchCompensators() using the accessToken obtained at point 6.1	CR			**Optional if you decide to perform step 6.5 or 6.6**
6.5	Obtain the compensator's details executing the command searchCompensatorsByInsurerIndex() using the acessToken obtained at point 6.1	CR			**Optional if you decide to perform step 6.4 or 6.6**

6.6	Obtain the compensator's details executing the command getOrganisation() using the acessToken obtained at point 6.6	CR			**Optional if you decide to perform step 6.4 or 6.5**
6.7	Obtain the hospital details executing the command getHospitalsList() using the accessToken obtained at point 6.1	CR			
6.8	Create a claim by executing the command addClaim() using the accessToken obtained at point 6.1.	CR	The claim is submitted into the system		
6.9	Get the list of claims executing the command getClaimsList() using the accessToken obtained at point 6.2	CR/COMP	CR: The claim is not in the claims list. COMP: The claim is in the claims list		
6.10	Allocate the claim to another branch executing the command allocateClaimToBranch() using the accessToken obtained at point 6.2	COMP			**This step is optional.**
6.11	Ask for a new accessToken and a new refreshToken executing the command refreshToken() including the refreshToken obtained at point 6.1	CR	The system returns a new accessToken and a new refreshToken for the claimant representative		
6.12	Ask for a new accessToken and a new refreshToken executing the command refreshToken() including the command refreshToken obtained at point 6.2	COMP	The system returns a new accessToken and a new refreshToken for the compensator		
6.13	Acknowledge the claim executing the command acknowledgeClaim() using the accessToken obtained at point 6.12	COMP			
6.14	Reject the claim to CR executing the command rejectClaimToCR() using the token obtained at point 6.12	COMP			
6.15	Get the list of notifications executing the getNotificationsList() using the command accessToken obtained at point 6.11.	CR			**Do not provide the A2ANotification flag/send the A2ANotification flag set to false**

6.16	Acknowledge the claim rejection executing the command acknowledgeRejectedClaim() using the accessToken obtained at point 6.11.	CR			
6.17	Resend the claim to the compensator executing the command resendRejectedClaim() using the accessToken obtained at point 6.11.	CR			
6.18	Get the claim details executing the command searchClaim() using the access token obtained at point 6.12	COMP			
6.19	ACKNOWLEDGE the claim executing the command acknowledgeClaim() using the accessToken obtained at point 6.12	COMP			
6.20	ACCEPT the claim executing the command acceptClaim() using the accessToken obtained at point 6.12	COMP			
6.21	Attach a note to the claim executing the command addAttachment() using the accessToken obtained at point 6.12	COMP			
6.22	Ask for a new accessToken and a new refreshToken executing the command refreshToken() including the refreshToken obtained at point 6.11	CR	The system returns a new accessToken and a new refreshToken for the claimant representative		
6.23	Ask for a new accessToken and a new refreshToken executing the command refreshToken() including the refreshToken obtained at point 6.12	COMP	The system returns a new accessToken and a new refreshToken for the compensator		
6.24	ACCEPT THE LIABILITY of the claim executing the command sendLiabilityDecision() using the accessToken obtained at point 6.23	COMP			
6.25	Acknowledge the liability admitted executing the command acknowledgeLiabilityAdmitted() using the accessToken obtained at point 6.22	CR			
6.26	Set INTERIM PAYMENT NEEDED = TRUE executing the command setInterimPaymentNeeded() using the accessToken obtained at point 6.22	CR			

6.27	Attach a note to the claim executing the command addAttachment() using the accessToken obtained at point 6.22	CR			
6.28	Perform an interim payment request executing the command addInterimSPFRequest() using the access token obtained at point 6.22	CR			
6.29	Execute a getClaim() to check the details of the interim payment request, using the accessToken obtained at point 6.23	COMP			
6.30	Attach a file to the claim executing the command addAttachment() using the accessToken obtained at point 6.23	COMP			**The file must be a dummy txt file containing only a line of text.**
6.31	REJECT the interim payment request executing the command rejectInterimSettlementPack() using the accessToken obtained at point 6.23	COMP			
6.32	Acknowledge the rejection of the interim settlement pack executing the command acknowledgeRejectedInterimSettlementPack() using the accessToken obtained at point 6.22	CR			
6.33	Set INTERIM PAYMENT NEEDED = TRUE executing the command setInterimPaymentNeeded() using the accessToken obtained at point 6.22	CR			
6.34	Ask for a new accessToken and a new refreshToken executing the command refreshToken() including the refreshToken obtained at point 6.22	CR	The system returns a new accessToken and a new refreshToken for the claimant representative		
6.35	Ask for a new accessToken and a new refreshToken executing the command refreshToken() including the refreshToken obtained at point 6.23	COMP	The system returns a new accessToken and a new refreshToken for the compensator		
6.36	Perform an interim payment request executing the command addInterimSPFRequest() using the access token obtained at point 6.34	CR			

6.37	Execute a getClaim() to check the details of the interim payment request, using the accessToken obtained at point 6.35	COMP			
6.38	Extend the interim payment decision timeout executing the command extendInterimPaymentDecisionTimeout() using the access token obtained at point 6.35	COMP			
6.39	Respond to the Interim payment request executing the command addInterimSPFResponse() using the accessToken obtained at point 6.35	COMP			
6.40	Execute a getClaim() to check the details of the interim payment request, using the accessToken obtained at point 6.34	CR			
6.41	Start the stage 2.1 executing the command setStage21Payment() using the accessToken obtained at point 6.34	CR			
6.42	Set INTERIM PAYMENT NEEDED = FALSE executing the command setInterimPaymentNeeded() using the accessToken obtained at point 6.34	CR			
6.43	Perform the S2SPF request executing the command addStage2SPFRequest() using the accessToken obtained at point 6.34	CR			
6.44	Ask for a new accessToken and a new refreshToken executing the command refreshToken() including the refreshToken obtained at point 6.34	CR	The system returns a new accessToken and a new refreshToken for the claimant representative		
6.45	Ask for a new accessToken and a new refreshToken executing the command refreshToken() including the refreshToken obtained at point 6.35	COMP	The system returns a new accessToken and a new refreshToken for the compensator		
6.46	Execute a getClaim() to check the details of the S2SPF payment request, using the accessToken obtained at point 6.45	COMP			

6.47	Extend the S2SPF decision timeout executing the command extendStage2SPFDecisionTimeout() using the access token obtained at point 6.45	COMP			
6.48	Respond to the S2SPF Request executing the command addStage2SPFResponse() using the token obtained at point 6.45	COMP			
6.49	Execute a getClaim() to check the details of the S2SPF payment response, using the accessToken obtained at point 6.44	CR			
6.50	Set S2SPF COUNTER OFFER NEEDED = YES executing the command setStage2SPFCounterOfferNeeded() using the accessToken obtained at point 6.44	CR			
6.51	Extend the S2SPF counter offer decision timeout executing the command extendStage2SPFCounterOfferTimeout() using the access token obtained at point 6.44	CR			
6.52	Perform the S2SPF counter offer executing the command addStage2SPFCounterOfferByCR() executing the accessToken obtained at point 6.44	CR			
6.53	Get the list of notifications executing the getNotificationsList() using the command accessToken obtained at point 6.45.	COMP			**Do not provide the A2ANotification flag/send the A2ANotification flag set to false**
6.54	Extend the S2SPF counter offer decision timeout executing the command extendStage2SPFCounterOfferTimeout() using the access token obtained at point 6.45	COMP			
6.55	Respond to the counter offer request executing the command addStage2SPFCounterOfferByCM() using the accessToken obtained at point 6.45	COMP			
6.56	Ask for a new accessToken and a new refreshToken executing the command refreshToken() including the refreshToken obtained at point 6.44	CR	The system returns a new accessToken and a new refreshToken for the claimant representative		

6.57	Ask for a new accessToken and a new refreshToken executing the command refreshToken() including the refreshToken obtained at point 6.45	COMP	The system returns a new accessToken and a new refreshToken for the compensator		
6.58	Attach a file to the claim executing the command addAttachment() using the accessToken obtained at point 6.56	CR			**The file must be a dummy txt file containing only a line of text.**
6.59	Set S2SPF COUNTER OFFER NEEDED = NO executing the command setStage2SPFCounterOfferNeeded() using the accessToken obtained at point 6.56	CR			
6.60	Set S2SPF AGREEMENT DECISION = FALSE executing the command setStage2SPFAgreementDecision using the accessToken obtained at point 6.56	CR			
6.61	Add the Court Proceeding pack request executing the command addCPPFRequest() using the accessToken obtained at point 6.56	CR			
6.62	Get the printable documents list executing the command getPrintableDocumentsList() using the accessToken obtained at point 6.57	COMP			
6.63	Get the printable document executing the command getPrintableDocuments() using the accessToken obtained at point 6.57	COMP			
6.64	Add the Court proceeding pack response executing the command addCPPFResponse() using the accessToken obtained at point 6.57	COMP			
6.65	Get the claim details executing the command searchClaim() using the access token obtained at point 6.56	CR			
6.66	Get the printable documents list executing the command getPrintableDocumentsList() using the accessToken obtained at point 6.56	CR			
6.67	Get the printable document executing the command getPrintableDocuments() using the accessToken obtained at point 6.56	CR			

6.68	Get the attachment list executing the command getAttachmentsList() using the accessToken obtained at point 6.56	CR			
6.69	Download an attachment executing the command getAttachment()using the accessToken obtained at point 6.56	CR			
6.70	Get the attachment list executing the command getAttachmentsList() using the accessToken obtained at point 6.57	COMP			
6.71	Download an attachment executing the command getAttachment()using the accessToken obtained at point 6.57	COMP			
6.72	Acknowledge the Court Proceeding Pack response executing the acknowledgeCPPFResponse() using the accessToken obtained at point 6.56	CR			

7.   Test Plan – Error Handling

In order to test the new error handling, it is sufficient to try to perform some of the steps described in the paragraph Test Plan – ELPL – Complete workflow in an incorrect way:

Nr.	Description	Who?	Expected results	Checks	Notes
7.1	Execute the step 6.1 with invalid credentials	CR	The system returns a login failure		
7.2	Execute the step 6.2 with invalid credentials	COMP	The system returns a login failure		
7.3	Execute the step 6.9 with an invalid token, both for CR and COMP	CR/COMP	The system returns a Bad Token error message		
7.4	Execute the step 6.29 with an incorrect applicationID	COMP	The system returns an "ApplicationID invalid" error message		
7.5	Execute the step 6.46 with an incorrect applicationID	CR	The system returns an "ApplicationID invalid" error message		

7.6	Execute the step 6.36 submitting an incorrect request, e.g. missing PSLA losses	CR	The system returns a "The process has followed an unexpected path", the response includes also the details tag containing more details about the error, e.g. in case of missing PSLA losses: *"<?xml version="1.0" encoding="UTF-8"?><ErrorData><ErrorList><Error Code="UNDEFINED" ErrorMessageCode="PSLALossesNeeded" FieldName="PSLA Losses" FieldsetName="PSLA Losses" Severity="Fatal" Type="Business"/></ErrorList></ErrorData>"*		
7.7	Execute the step 6.39 submitting an incorrect request, e.g. missing Cru field	COMP	The system returns a "The process has followed an unexpected path", the response includes also the details tag containing more details about the error, e.g. in case of missing CRU field: *"<?xml version="1.0" encoding="UTF-8"?><ErrorData><ErrorList><Error Code="UNDEFINED" ErrorMessageCode="CRUReferenceNumberOrCRUComment" FieldName="CRU Reference Number" FieldsetName="CRUReference" Severity="Fatal" Type="Business"/></ErrorList></ErrorData>"*		
7.8	Execute step 6.20 before step 6.19	COMP	The system returns a "The operation could not be applied in this point" error.		
7.9	Execute step 6.17 before step 6.16	CR	The system returns a "The operation could not be applied in this point" error.		
7.10	Perform one of the above refreshToken() with incorrect credentials	CR/COMP	The system returns a Bad Token error message		

8.   Need Help?

If you have any questions related to any of the A2A documentation or the A2A token service in the integration/test site, complete the form "A2A Integration/Testing Contact Us - A2A users / Software Houses" available on the Claims Portal web site https://www.claimsportal.org.uk/contact-us/ attach to an email and send to the Help Desk, email address below. Please provide all information requested if information is missing it may create a delay or result in us being unable to assist you.

SEND TO: helpdesk@rapidclaimsettlement.org.uk
SUBJECT: A2A Integration Testing or Change password expiry date (whichever is applicable)