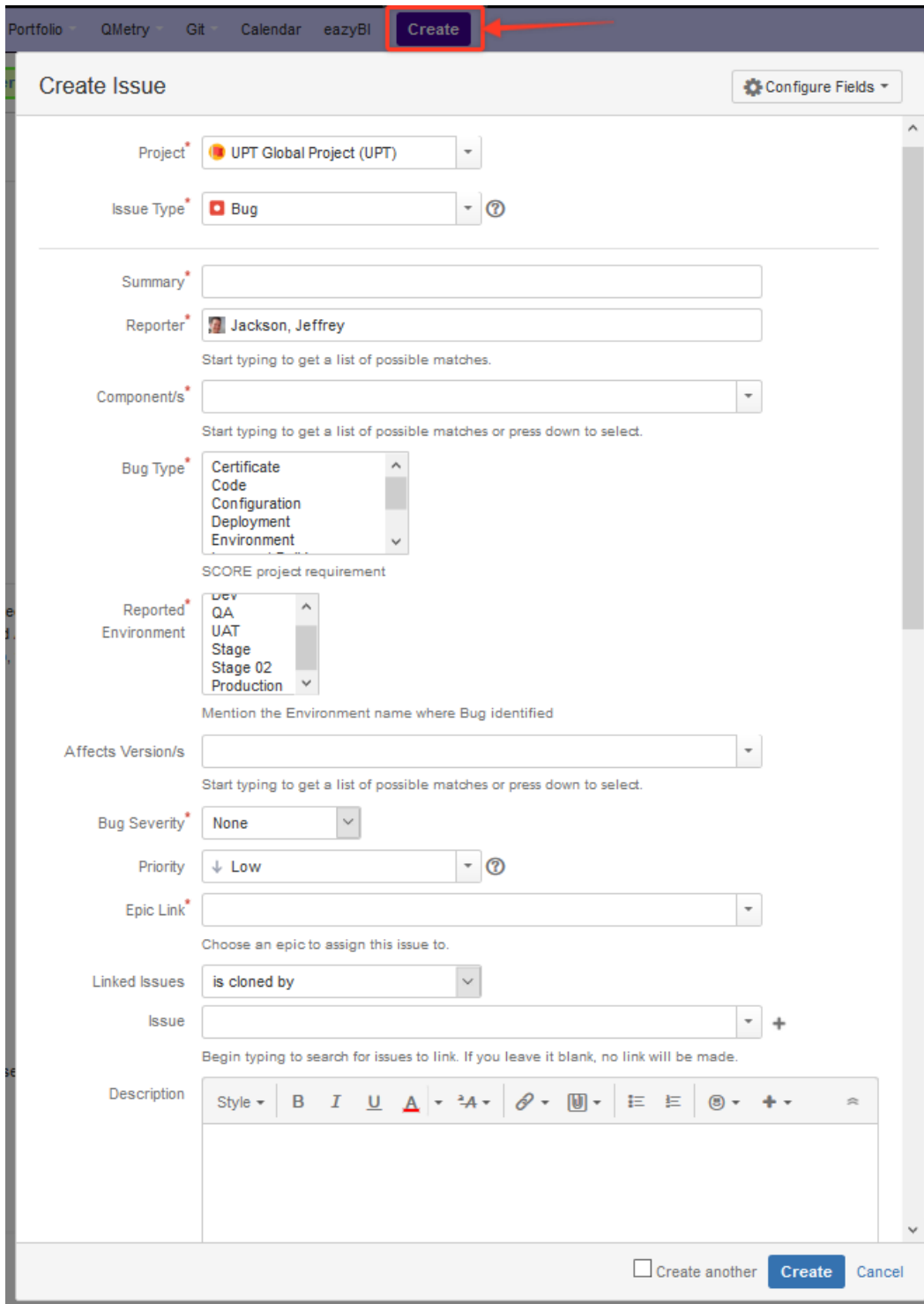


Creating a Bug in Jira

In Jira, Bugs can be created from the purple 'Create' button on the top navigation bar or while executing a test case on the Test Execution screen. This document will create the Bug from the top navigation bar, where it can be access by all user, on any screen.

1. Initial Creation

- Select the purple 'Create' button at the top of the Jira navigation bar. This will populate the 'Create Issue' window for you to fill out the details on the Bug that was discovered.



The screenshot shows the Jira 'Create Issue' form. At the top, a red box highlights the 'Create' button in the navigation bar, with a red arrow pointing to it. The form itself is titled 'Create Issue' and includes a 'Configure Fields' button. The fields are as follows:

- Project:** UPT Global Project (UPT)
- Issue Type:** Bug
- Summary:** (empty text field)
- Reporter:** Jackson, Jeffrey
- Component/s:** (empty dropdown menu)
- Bug Type:** Certificate (dropdown menu with options: Certificate, Code, Configuration, Deployment, Environment)
- Reported Environment:** Dev (dropdown menu with options: Dev, QA, UAT, Stage, Stage 02, Production)
- Affects Version/s:** (empty dropdown menu)
- Bug Severity:** None (dropdown menu)
- Priority:** Low (dropdown menu)
- Epic Link:** (empty dropdown menu)
- Linked Issues:** is cloned by (dropdown menu)
- Issue:** (empty dropdown menu)
- Description:** (rich text editor with formatting options)

At the bottom of the form, there is a checkbox for 'Create another' and two buttons: 'Create' and 'Cancel'.


2. Explanation of Fields

Fields that should always be filled out with information are marked as required, with a red asterisk. All required fields must be filled out before the blue 'Create' button is selected, to create the Bug. There are also other fields that should be filled out on a Bug, that are not marked as required. The information for these fields may not be initially available, may be filled out as the Bug progresses through each status, or may not be relevant for each team. Always provide as much information as possible, to make sure that the Bug can be triaged and resolved correctly, and efficiently.

- **Project (Mandatory):** This is the project that your application is assigned too.
- **Issue Type (Mandatory):** Always select Bug.
- **Summary (Mandatory):** This is a high-level explanation of the issue. When reading this, another user should have a general understanding of the issue and the application(s) affected.
 - **Naming Convention:** [Application][Operating System][Browser] – [Project or Function] – Explanation of Issue.
 - **Examples:** “MS-Silverpop - MFO - Final Order Confirmation email is not being sent”; “UO iOS – Wallet – Cannot update my primary credit card”; “UO.com Chrome – ICE – Delivery Methods are not loading in the Cart”
- **Reporter (Mandatory):** User that found the Bug. It is important that this is correctly filled out, in case anyone has questions on the issue or needs additional details.
- **Component/s (Mandatory):** Team that is responsible for fixing the issue. More than one team can be added.
- **Bug Type (Mandatory):** What kind of Bug and what the contributing factor is. This is a multi-select window.
- **Reported Environment (Mandatory):** When the issues was found and/or where it currently exists. This is a multi-select window.
- **Affects Version/s (Mandatory):** This states what version or release of the application that the Bug was found it. This is important for triage and reproduction, as well as used in other department reporting, like Release Management. Though, this is not a required field, it is expected to be filled out. More than one value can be added.
- **Bug Severity (Mandatory):** This explains how critical/important the Bug is.
- **Priority:** This also describes the level of importance or urgency for resolving the issue. The Bug reported is expected to fill this out initially, but it may be altered by the Product Owner or Application Owner.
- **Epic Link (Mandatory):** This determines the project or charter that this Bug is affecting. More than one value can be added.
- **Linked Issues:** This dropdown explains why this Bug is being linked to another Jira issue.
 - **Issues:** This is where the intended linked Jira issue(s) is added.
- **Description (Mandatory):** This gives a full explanation of the Bug and all the details needed to understand and reproduce the issue. Even though it is not an initially required field, it is required to be fully filled out before any future work can be done to fix the Bug.
- **Labels:** If used by a team, labels are a searchable field used for filtering or categorizing a Jira issue.
- **Attachment (Mandatory):** The section of a Bug is where you can visually explain and show the issue that was discovered. Even though it is not an initially required field, attachments showing the Bug are required before any future work can be done to fix the Bug.
- **Assignee:** This is who owns the Bug, in its current state/status.
- **Dev Owner:** This is for the developer that is assigned to fix the Bug. This may not be known until the assigned feature team commits to the work.
- **QA Owner:** This is for the QA resource that is assigned to retest the Bug. This may not be known until the assigned feature team commits to the work.
- **UAT Owner (if applicable):** This is for the business resource that is assigned to retest the Bug. This may not be known until the assigned feature team commits to the work.
- **Product Owner:** This is for the Product Owner of the application that is being fixed. This may not be known until the bug is appropriately triaged.

- **Program Increment:** This will be filled out when a team commits to the work and knows when they are starting their work. This is usually filled out by the Scrum Master.
- **Story Points#:** Only applicable if the scrum team uses story points for Bugs. This would be filled out by the Scrum Master.
- **Original Estimate:** Not applicable in our current Scrum process.
- **Remaining Estimate:** Not applicable in our current Scrum process.
- **Sprint:** This will be filled out when a team commits to the work and knows when they are starting their work. This is usually filled out by the Scrum Master or Product Owner.
- **Fix Version/s:** This is the version or release of the application in which the Bug was resolved in or no longer presented itself in.
- **External Issue ID:** Used for unique identifying numbers of outside application, specifically Service Now Incident numbers. This is usually populated by Product Owners, Business Owners, or Release Management.
- **External ID Date:** Date of when the external issue (ticket) was created.
- **Applications (Mandatory):** This explains what application needs to be fixed when resolving the Bug. Though this is not a required field, it is important for issue searching and reporting.



3. Bug Details


UPT Global Project / UPT-13575

Services MS-Silverpop - MFO - Final Order Confirmation email is not being sent

Edit
Comment
Assign
More ▾
Begin Triage

Details

Type:	 Bug	Status:	TO DO (View Workflow)
Priority:	 Normal	Resolution:	Unresolved
Affects Version/s:	UO Android Mobile App 1.20.0, UO iOS Mobile App 1.20.0	Fix Version/s:	None
Component/s:	Mobile Core	Security Level:	Team+Reporter+Assignee+Watcher(s)
Labels:	None		
Charter:	Mobile Run		
Value Stream:	Integrated Content & Commerce		
Bug Type:	Code		
Reported Environment:	UAT		
Bug Severity:	Minor Problem		
Epic Link:	P19 Mobile Regression Defects		
Applications:	ms-utilities-silverpop		

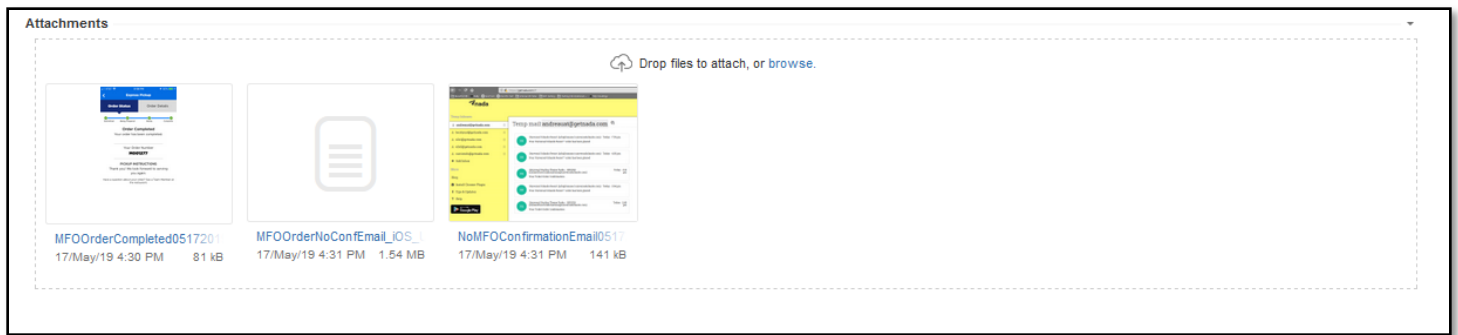
- The Bug Details sections (including the Summary line above) is intended to give users a snapshot of:
 - What the problem is
 - What application(s) is effected
 - Which environment(s) the Bug was found in
 - What team owns the Bug for resolution
- There are other important fields that assist with triage and explanation of the issue, but these are key details that will give other users the appropriate mindset for resolving the Bug.

4. Bug Description

<p>Description</p> <p>In the Mobile Food Ordering flow, the user is not receiving the final order confirmation email after the order is completed. The user is successfully receiving other UO emails (ie Account Creation, Add Payment Method, and MFO Order Submitted), but not the MFO Order Confirmation email. This issue has occurred when making orders through iOS and Android, in UAT, but the issue does not present itself in PROD with the 1.20.0 Test Flight build for iOS. The issue presented itself over 5 times today and no known emails have been received today (MFO Confirmation Emails). Also, the orders and the payments of all of the test transactions today are displaying in Cloudant; example orders M001271, M001272, M001277.</p> <p>Devices used in UAT: iPhone X (12.3) & Galaxy S8 (8.0)</p> <p>Steps to Reproduce:</p> <ol style="list-style-type: none"> 1. Download the 1.20.0 Next Release for UAT, on an iOS or Android Device. 2. Login with a registered user that has a valid credit card and accessible email account. 3. Open the left navigation and select Order Food and Drinks. 4. Select a Venue and a Food Item within that Venue. 5. Add the Food Item to your Cart. 6. Proceed to your Cart. 7. Proceed to the Checkout screen. 8. Place the Order. 9. Select the Prepare My Order button. 10. In Venue Next's Canopy application, move the Order to Ready and then to Complete. 11. Navigate to the email inbox of the registered user used to place the order. <p>Expected Results:</p> <ul style="list-style-type: none"> • There should be 2 MFO emails, an Order Submitted email and an Order Confirmation email for the MFO purchase. <p>Actual Results:</p> <ul style="list-style-type: none"> • There is only 1 email, the Order Submitted email. There is no Order Confirmation email. <p>Please see the screen shots and Charles Trace attached below.</p>

- The Description field give a full explanation of the Bug and all the details needed to understand and reproduce the issue. Within the Description field, there are key elements that should always be included, if applicable.
 - **Description/Explanation of the issue:** Within a few sentences, the reporter should be able to fully explain the issue that was discovered.
 - **Environment:** This explains what environment(s) the issue was found in and/or what environment(s) the issue exists in.
 - **Devices/Browsers used:** If applicable, the web browser, mobile device, and/or operating system should be stated.
 - **Username/Passwords:** List any username and password used to recreate the Bug, as the issue may be specific to the data associated to that user.
 - **Confirmation Numbers:** If the issue presents itself after a purchase is made, then the order confirmation number will need to be provided. This way the development team can see the specific issue that you are reporting.
 - **Steps to Reproduce:** This section is very important! All Bugs reported MUST include steps to reproduce the issue. Not everyone looking at the Bug will know the application as well as the reporter, or even know where in the application the reporter was, when the Bug presented itself. Steps to reproduce should be very detailed, so that new or experienced users on the application could follow them to reproduce the issue.
 - **Expected Results:** This is what the user is expecting to see if there was no Bug presented.
 - **Actual Results:** This is what the user is actually seeing, due to the Bug presenting itself.
- If there is any additional information available to explaining the issue, then it should be added.

5. Attachments



- The Attachment section of a Bug is where you can visually explain and show the issue that was discovered.
- Examples of important attachments:
 - Application Screen Shots
 - Charles Trace
 - Application Logs
 - Application Payloads (either used for mocking a call or the actual payload that shows the bug)
 - Screen Recording/Video
- The more information that is available to the team, the easier it will be to reproduce and resolve the Bug.
- Per QE standards, screen shots or video, and a Charles trace is required when submitting a Web or Mobile Bug.

6. Bug Example

NBCUniversal

Dashboards

Projects

Issues

Tempo

Capture

Boards

Tests

Portfolio

QMetry

Git

Calendar

easyBI

Create

need any help with Jira or Confluence, please don't hesitate to raise a ticket here: <https://jira.inbcu.com/servicedesk/customer/portal/161>

UPT Global Project / UPT-13575

Services MS-Silverpop - MFO - Final Order Confirmation email is not being sent

Edt

Comment

Assign

More

Begin Triage

Details

Type:Bug

Priority:Normal

Affects Version/s:UO Android Mobile App 1.20.0, UO iOS Mobile App 1.20.0

Component/s:Mobile Core

Labels:None

Charter:Mobile Run

Value Stream:Integrated Content & Commerce

Bug Type:Code

Reported Environment:UAT

Bug Severity:Minor Problem

Epic Link:PI9 Mobile Regression Defects

Applications:ms-utilities-silverpop

Status:TO DO (View Workflow)

Resolution:Unresolved

Fix Version/s:None

Security Level:Team+Reporter+Assignee+Watcher(s)

People

Assignee:Jeremy Scheinberg

Reporter:Jackson, Jeffrey

Watcher(s):

Votes:0

Watchers:Start watching this issue

Dates

Created:17/May/19 4:32 PM

Updated:09/Sep/19 4:01 PM

> Summary Panel

> Collaborators

TFS Check-ins

7 Repos

0 Check-ins

Agile

View on Board

Description

In the Mobile Food Ordering flow, the user is not receiving the final order confirmation email after the order is completed. The user is successfully receiving other UO emails (ie Account Creation, Add Payment Method, and MFO Order Submitted), but not the MFO Order Confirmation email. This issue has occurred when making orders through iOS and Android, in UAT, but the issue does not present itself in PROD with the 1.20.0 Test Flight build for iOS. The issue presented itself over 5 times today and no known emails have been received today (MFO Confirmation Emails). Also, the orders and the payments of all of the test transactions today are displaying in Cloudant, example orders M001271, M001272, M001277.

Devices used in UAT: iPhone X (12.3) & Galaxy S8 (8.0)

Steps to Reproduce:

- Download the 1.20.0 Next Release for UAT, on an iOS or Android Device.
- Login with a registered user that has a valid credit card and accessible email account.
- Open the left navigation and select Order Food and Drinks.
- Select a Venue and a Food Item within that Venue.
- Add the Food Item to your Cart.
- Proceed to your Cart.
- Proceed to the Checkout screen.
- Place the Order.
- Select the Prepare My Order button.
- In Venue Next's Canopy application, move the Order to Ready and then to Complete.
- Navigate to the email inbox of the registered user used to place the order.

Expected Results:

- There should be 2 MFO emails, an Order Submitted email and an Order Confirmation email for the MFO purchase.

Actual Results:

- There is only 1 email, the Order Submitted email. There is no Order Confirmation email.

Please see the screen shots and Charles Trace attached below.

Test Cases/Acceptance Criteria

Search By Key/Summary

+

Add

Reuse

#	Key	Summary	Created By
No data available			

Test Runs

View Test Run

OneDrive attachments

Sign in to attach from OneDrive

Attachments

Drop files to attach, or browse.

MFOOrderCompleted051720

17/May/19 4:30 PM

81 kB

MFOOrderNoConEmail_IOS

17/May/19 4:31 PM

1.54 MB

NoMFOConfirmationEmail051

17/May/19 4:31 PM

141 kB