Quality Engineering
QA Testing Process and Standard

Document Location

The most up to date copy of this document can find at the following Wiki:
https://wiki.inbcu.com/display/QECD/Quality+Engineering

All Project at Universal

Universal Parks & Resorts has embarked on an initiative to enhance the overall guest & team member experience. This initiative focused on provided guests with a more seamless and tailored theme park resort vacation allowing guests to plan and experience their dream vacation effortlessly. All projects will enable an integrated business application and guest data platform that will provide a holistic, granular view of UPR's prospective, current, and past guests. An initiative will provide a full guest lifecycle experience management and real-time analytics.

Agile Delivery

- All DEP project follows the Agile Software Delivery Methodology. At the heart of Agile is the idea of continuous delivery with fully communicating cross-functional teams. Below are some standard Agile terms used across all projects:
- **Sprint/Iteration:** A theme-driven timebox of requests to be worked on and accepted within a release of a product; it is defined in an iteration planning meeting and is completes with an iteration demo and review meeting. The terms iteration and sprint are used synonymously
- **Product Backlog:** A collection of prioritized requests for work to be done
- **Backlog Grooming:** An ongoing process whereby the product owner or customer manages the product backlog based on information gathered in the feedback cycles inherent to agile practices.

The activities of backlog grooming can include: adjusting rank; breaking down stories that are going to be worked on in the next few iterations; creating new stories; updating existing stories; deleting obsolete stories; elaborating acceptance criteria.

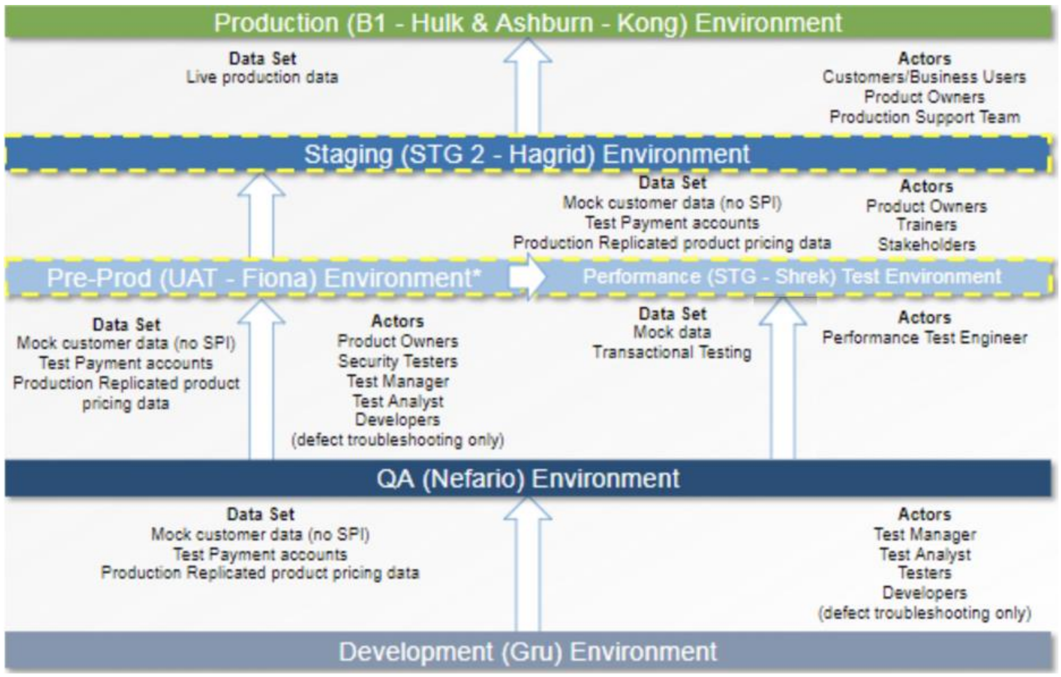
- **Scrum Master:** The mom in the kitchen, focus and decision consensus building, blocks and tackles obstacles for the team, escalates issues
- **Product Owner:** A role on an agile delivery team that is responsible for collecting and ranking business requirements on the product backlog. A product owner does not manage a delivery team but communicates what must be built in the next release or iteration. In exchange for the team's commitment to finish the top-most ranked work in an iteration, the product owner agrees to protect the team from any changes in requirements during the iteration
- **Release**: Multiple sprints make up a release (varies)
- **PI:** Program Increments, means the same thing as Releases (used in more scaled Agile)

In Agile, a project operates and is divided into different levels of delivery:

- **Epic(Rally)/Initiatives(JIRA):** Business processes and customer experiences utilized by Universal end users, park team members, operations members, and business resources
- **Features(Rally)/Epic(JIRA):** Entire application components that makeup business processes and customer experiences
- **User Stories(Rally)/Story(JIRA):** Individual application components, when strung together, they create Epic.

Environments:

Environment Structure


Production (B1 - Hulk & Ashburn - Kong) Environment
Data Set
Live production data
Actors
Customers/Business Users
Product Owners
Production Support Team
Staging (STG 2 - Hagrid) Environment
Data Set
Mock customer data (no SPI)
Test Payment accounts
Production Replicated product pricing data
Actors
Product Owners
Trainers
Stakeholders
Pre-Prod (UAT - Fiona) Environment*
Performance (STG - Shrek) Test Environment
Data Set
Mock customer data (no SPI)
Test Payment accounts
Production Replicated product pricing data
Actors
Product Owners
Security Testers
Test Manager
Test Analyst
Developers
(defect troubleshooting only)
Data Set
Mock data
Transactional Testing
Actors
Performance Test Engineer
QA (Nefario) Environment
Data Set
Mock customer data (no SPI)
Test Payment accounts
Production Replicated product pricing data
Actors
Test Manager
Test Analyst
Testers
Developers
(defect troubleshooting only)
Development (Gru) Environment

Environment Details:

Dev (Gru): Environment utilized by Development Teams to add and modify their code. Developers will code on their local, point to the Dev environment, and deploy to Dev for unit testing. Unit testing is tracked a "Task" level on User Stories; also Environment utilized by the QA team to test User Stories during the Sprint test phase as well as Product Owners accept User Stories against this Environment.

Dev Prime (Bumblebee): Environment that utilized for testing and comparing architectural components. Note: The Dev Prime environment is not a part of the standard development and testing process for releases nor is it a full-stack environment

QA (Nefario): Environment utilized by the QA team to test User Stories during the QA test phase. The Regression and Integration test phases also are executed in the QA environment as well as Product Owners accept User Stories against this Environment.

UAT (Fiona): Environment utilized by the E2E team to execute end to end test flows during the E2E test phase. The UAT test phase also takes place in here for the business's acceptance of the release to promote into Production. The use of price data and product configuration replicated from Production allows for verification of proper values. Defects found in UAT need to be re-tested in the QA environment before promoting up to UAT

Staging (Shrek): Environment utilized by the Performance test team to perform load and sale tests during the Performance test phase. The use of price data and product configuration replicated from Production allows for verification of proper values. Defects found in STG need to be re-tested in the QA environment before promoting up to STG

Staging 2 (Hagrid): Environment that mirrors Production and is utilized to test Hotfixes

B1 Data Center (Hulk): Production for Volcano Bay data. Each release to PROD will require a post-implementation "Smoke test" performed by Product Owners. The Production Support team is responsible for resolving defects found in PROD

Ashburn (Kong): Environment that lives in Production for digital (mobile and web) channels. Each release to PROD will require a post-implementation "Smoke test" performed by Product Owners. The Production Support team is responsible for resolving defects found in PROD
Note: We moved all data to B1 two weeks ago, only commerce data in Ashburn data center.

JIRA Management
JIRA is the DEP projects tool utilized to manage Initiatives, Epics, Stories, and Bugs. Qmetry is JIRA plug-in used to track test execution progress and test management. To gain access to JIRA, contact Amber Jackson (amber.jackson@universalorlando.com).

Testing Methodology

Test planning, preparation, and execution represent the essential activities involved in the testing effort to ready 'Universal's DEP project for Production. The DEP project testing methodology consists in testing new features and functions through four stages for each project phase: Dev-

QA / System Testing (Story), Integration Testing (Epic), User Acceptance Testing (UAT) (Initiatives and Epic), and Performance Testing (Load and Scale).

Test Phases

Dev-QA / System Test Phase (Story)

The QA / System Test Phase is executed in a sprint along with Development and Product Owner acceptance at the Story level.
This test phase:
1) Verifies the proper execution of individual application components and
2) Checks the integration of applications including internal and external interfaces, with their crucial hardware, software, and infrastructure components. Testing tasks, test scenarios are created, managed, and completed within each Story in JIRA for the appropriate release. The QE Team is responsible for this phase of execution.
Note: QA can create a bug in System Test phase when the issue is related to other team and blocking your Story to complete work in a sprint or running a regression in QA for accepted Story.

QA-Integration Test Phase (Epic)

The Integration Test Phase is kicked off after the Product Owners have accepted all Stories, and the code has been locked down.
This test phase:
1) Verifies proper execution of the entire application components including interfaces to other applications and
2) Executes end-to-end testing of business processes and customer experiences. Test scenarios and Bugs are created, managed and completed within the Test Plan function in JIRA for the applicable release. The QE Team is responsible for this phase of execution.

E2E / UAT Test Phase (Initiatives and Epic)

The E2E / UAT Test Phase is kicked off after the Integration test phase is complete, and the code has been locked down.
This test phase:
1) Executes end-to-end business processes and customer experiences by Universal end-user and Product Owner resources to demonstrate system functionality and performance and
2) Accepts end-to-end business processes and customer experiences as design, built and tested before Production. Test scenarios and defects are created, managed, and completed within the Test Plan function in JIRA for the appropriate release. The E2E / UAT Team is responsible for this phase of execution.

Performance Test Phase (Load and Scale)

The Performance Test Phase is kicked off the Integration test phase is complete, and the code has been locked down. This test phase runs in parallel with the QA/System Phase Test Phase. This test phase:
1) Verifies the DEP project architecture is sufficient to support the highest expected volume of product users (I.e., Load and scalability) and

2) Determines if the anticipated system response time (performance) is adequate for DEP business functionality and customer usability. Test scenarios and bugs are created, managed, and completed within the Test Plan function in JIRA for the appropriate release. The Performance Test Engineers are responsible for this phase of execution.

Regression Test Phase

The Regression test phase verifies an application that has been modified has not produced a negative impact on previously working functionalities.
This phase is kicked off:
A) During the QA / System or Integration test phases of the Implementation Stage introducing the new code (I.e., New code touching existing DEP or non-DEP functionality),
B) Before deploying the upgrade or patch to Production (I.e., A Code deploys an update or patch after the Implementation Stage has completed) or
C) Before implementing the fix or changes to Production (I.e., Fixes to the DEP Solution or changes to legacy applications). Test scenarios/Test run and bugs are created, managed, and completed within the Qmetry in JIRA for the appropriate release. The QE and Production Support Team is responsible for this phase of execution.

Entry and Exit Criteria

Entry and exit criteria are critical milestones that must be met to transition throughout the testing phases. Entrance criteria are those factors that must be present, at a minimum, to be able to start an activity. Exit criteria are those factors that must be present to declare an activity completed.

Test Phases / Activities	Entry Criteria	Exit Criteria	Owner(s)
QA / System Test (conducted as part of the path to Production)	-Test environment ready -Completed Unit Test -Compile and link code without errors -Completed QA / System Test Plan -Completed QA / System Test scenarios and scripts -Completed QA / System Test Execution Schedule -Required data available	-No open Sev1 defects -Less than 10 High Priority Sev2 bugs with determined resolution unless approved by key stakeholders -Prioritization of Sev3 bugs -Completion of one end-to-end testing cycle	QE Scrum Team
Integration Test (conducted as part of the path to Production)	-Completed QA / System Test -Compile and link code without errors -Required data available -Test environment ready -Completed Integration Test Plan -Completed Integration test scenarios -Completed Integration Test Execution Schedule	-Completion of designated passes -No open Sev1 bugs -Less than 10 High Priority Sev2 bugs with determined resolution unless approved by key stakeholders -Prioritization of Sev3 bugs -Stakeholders acceptance of open bugs	QE Scrum Team

E2E / User Acceptance Testing (UAT) (conducted as part of the path to Production)	-Completed Integration Test -Compile and link code without errors -Completion of code migration to UAT environment -UAT environment ready -Required data available -Completed UAT Test Plan -Completed UAT Test Scenarios -Completed UAT Test Execution Schedule	-Completion of designated passes -No open Sev1 bugs -Less than 10 High Priority Sev2 bugs with determined resolution unless approved by key stakeholders -Prioritization of Sev3 bugs -Stakeholders acceptance of open bugs	E2E QE team and UAT team (Product Owners)
Performance Test (conducted as needed)	-Completed Integration Test -Test scenarios selected -Performance requirements defined -Performance environment and data ready	-Service level requirements are met or defects are approved -Stakeholders acceptance of any exceptions	QE Team (Performance Test Engineers)
Regression Test (conducted as needed)	-Regression test scenarios selected -Test environments ready -Required data ready	-No unintended adverse effects found -Stakeholders acceptance of any exceptions	QE Team
Implementation Test (conducted as part of the path to Production)	-Completed UAT -Open defects are approved by key stakeholders -UAT Results Signed Off	-Stakeholders acceptance of any bugs	Implementation Team

Story Acceptance

Story acceptance is performed solely by the Product Owners. Product Owners will monitor Kanban and review stories in the "Ready for PO Review" column. They have the responsibility of signing off on User Story functionality and deeming the experience ready for Implementation.

- The Product Owners will test the functionality of each Story in the QA environment to ensure the functionality has been implemented as required
- If the Product Team does not find any bugs during their review and the functionality meets all their acceptance criteria as intended, the Story is formally moved to "Accepted"
- If the Product Team does find issues with how the functionality has been implemented and does not believe the requirements have been met, they will log a bug(s) against the Story for the development teams to triage
- In some cases, the Product Owners may need assistance from the testing team in order explain or recreate specific tests

Test Planning

During the Test planning activity, the QE Test Leads develop a test strategy which focuses on scope, schedule, assumption and risks / mitigation for the applicable release. During this timeframe, the QE Test Leads also determine the application and code components that must be available for testing.

Test Preparation

During the Test Preparation activity, the QE Testing team writes detailed test scripts for each test scenario to validate the Story requirements and exit criteria. The team also identifies test data needed for test execution in the QA environment.

Defining Requirements

Prior to creating test scenarios, the requirements and exit criteria of each Story need to be clearly defined and complete. Requirements are defined by the Business and Product Owners (see Appendix for teams). Through Solutioning and Backlog Grooming meetings, the Product Owners and Analysts are tasked with defining the solution into solidified requirements and exit criteria that document the desired experience for the end user. The Development teams code based off these requirements / exit criteria.

After the requirements are locked down and the Stories have the proper UI/UX materials (I.e. Comps and Copy Documents), Development teams will add "Dev Tasks" to the user stories to track their coding. During Sprint Planning, Scrum Masters will and pull Stories into a sprint for development and testing, based off business priority.

Note: While it would be best practice for requirements to be locked down on Day 1 of the sprint when Stories are pulled in for development, requirement often change during the sprint. When requirements change, the Product Owners and Analysts either highlight the impacted requirements in yellow (signifies a change) or in red (signifies removal).

UI/UX Materials - Comps, Copy Documents

Prior to Stories being deemed ready for development, the UI/UX materials need to be aligned and complete. Comps are the "templates" that developers reference to build the visual look and feel of the front-end application (I.e. Mobile, Web, TMT, ACP App). Originally a third-party vendor, Sapient, developed the comps, however they have since gone through many updates and refactors. For each Story we test, we refer to these documents as the "source of truth" for what the screens should look like to the end user.

The copy documents are developed to define the specific text and messaging throughout the front-end applications. This includes but is not limited to Product Names, Error Messaging, and Legal Text documents (see Appendix for documentation links).
As soon as the Story is aligned to the sprint, the QE team creates Test Cases for the Story. These Test Cases align exactly with a small portion of the requirements, giving full testing coverage on all elements of the expected functionality. Each test case lives within the Story, but are independent and each represent different functionality.

Testing Task Identification

On Day 1 of each sprint the QA team pulls down all Stories aligned to the sprint and applicable release as they have been accepted into sprint for development. The QE team analyzes each

Story to define which stories need to be tested. A testing task is aligned to each Story the team will execute against to track test execution progress. When execution kicks off the task moves from Defined to In Progress. When testing is complete and the Story is ready for Product Owner acceptance, the task moves from In Progress to Complete. Note: A testing task will move to Complete only if all test scenarios are Passed.

Test case Creation

The QA Test team is responsible for creating test cases against the requirements / exit criteria of each User Story that will be tested. The primary objective of testing is to identify and resolve defects prior to implementation. Test scenarios need to be written in a way that will ensure the right solution has been build and that the solution has been build right.

Prior to writing test cases, the Tester needs to have a complete understanding of what functionality the User Story is detailing. Requirements should not be copy and pasted from the User Story into the test cases. Instead, Testers should review the Stories for gaps and work with the Product Owners / Analysts to get all questions addressed. Having this understanding allows for the Tester to define edge cases that may not explicitly be written in the requirements.

Example: Test Case Creation:

- Requirement: When the guest taps Complete Purchase, the order is sent to Galaxy, and the guest is redirected to the Confirmation Screen.
- Test Case 1: Validate that when a guest taps Complete Purchase, that order makes it to Galaxy.
- Test Case 2: Validate that when a guest taps Complete Purchase, the guest is then redirected to the Confirmation Screen.

Example: Requirements Review:

- What kind of purchase should I be making to validate this process (I.e. Registered, unregistered, what product types, etc.)?
- What does sending to Galaxy mean? What exact information should be sent there as part of the order (I.e. OrderID, confirmation number, item purchased, etc.)
- Does the order interact with any other systems or undergo any further processing prior to reaching Galaxy?

Within each Story there is a subset of test cases that need to be included for QA test execution. These test cases consider the visual UI/UX of each screen per breakpoint, browser, and piece of hardware for Mobile, Web and TMT. Please find below the subset test cases to include during the QA Test Phase:

Mobile:

- Validate the UI/UX elements render properly and conform to provided wireframes/PSDs - Android
- Validate the UI/UX elements render properly and conform to provided wireframes/PSDs - iOS

Web:

- Breakpoint: Validate the UI/UX elements render properly and conform to provided wireframes/PSDs - Desktop, Mobile, Tablet (one test case is created per breakpoint)
- Browser: Validate functionality and design for all test scenarios - Chrome, Edge, Firefox, IE11, Safari (one test case is created per browser)

TMT:

- Validate the UI/UX elements render properly and conform to provided wireframes/PSDs - Desktop
- Validate the UI/UX elements render properly and conform to provided wireframes/PSDs – Tablet

Creating Test Cases for User Stories

After understanding the requirements and functionality that is documented in each Epic/Story, the tester must manually create the appropriate test cases to validate the functionality and UI/UX of the completed development work. Note: It is important that the Product Owners, Development teams and QA team have the same understanding of the intended functionality.

To successfully create a test case at the Story level:

- Check JIRA TRAINING DOCUMENT SECTION.

Test Step Definition

Test steps outline contain step-by-step actions of what is required to successfully execute a test case.
The Input is the action executed by the user and/or system and the Expected Result is what should happen after the action is executed.
Test steps should provide enough detail that would enable an end user that has not been a part of the development or test phase cycles to execute the test case from end-to-end.

These steps can be added during test case creation or by navigating to the Test Scenarios tab after adding the test cases to a Story or Test folder.

- Check JIRA TRAINING DOCUMENT SECTION.

Regression and Integration Test Plan Creation

The QA Test team is responsible for creating test scenarios for the Regression and Integration Test Phases. As we are testing at the Initiative and Epic level for these two test phases, we do not align our test scenarios to Stories. Instead we create Test Plans within JIRA to track our end-to-end testing of the customer experiences and business processes of the applicable release.

Test Execution

During the Test Execution activity, the QE Test team is responsible for executing test scenarios, documenting test results, and managing defect triage through closure. The QA Test Leads are responsible for leveraging JIRA to provide testing status reports as well as preparing CAB and/or eCAB documentation to gain approval for implementation.

Documenting Test Results

A test case cannot be Passed, Failed or Blocked in JIRA without formal test results. Test results act as evidence of testing all new functionality that will deploy to Production during implementation. Test results are a direct input into CAB and eCAB documentation as well as are important from an auditing perspective so it is imperative that each test case contains a result.

Passing Test Cases

Test cases are Passed when the desired outcome of the test case / requirement is achieved. When passing test cases, the tester must include a screenshot(s) as proof of the desired outcome as testing evidence. To pass a test case, make sure update each test steps with pass/fail/block:

Failing Test Cases

Test cases are Failed when the desired outcome of the test case / requirement is not achieved. When failing test cases, it is important to include the bug number for the issue found during test execution that did not align to the requirement. It is important to note that only one test case should be failed by a specific bug. All other test cases that are impacted by the bug test case will be placed into Blocked status.

Blocking Test Cases

Test cases are Blocked when a defect prevents the tester from being able to execute the test case / validate the requirement. When blocking test cases, it is important to include the reason the test case is being blocked, whether that be a defect or another cause. It is not sufficient to ever block test case without a bug listed in the test case or test step comment section.

Bug Management

Throughout test execution, issues will arise where the business requirements / exit criteria are not being met and the system is not functioning a desired. When this happens, bugs are raised in JIRA to track the resolution of the issue at hand. Bugs can also be raised for issues causing a poor end user experience (I.e. System slowness, visual miss-alignment, etc.). The QA Test team is responsible for bug management from the time a bug is raised, through triage, resolution and closure for the QA, Regression and Integration test phases.

Logging New Bugs

During the QA Test Phase, defects should be logged and aligned to Stories / Test Cases in JIRA. During the Regression and Integration Test Phases, bugs will be aligned to Test Cases.

When logging bugs, there are several fields that need to be populated for the bug to contain the proper information / documentation to allow for efficient and accurate to be tracking, triage and resolution. These fields include:

1. **Bug Name:** A reviewer should be able to understand the defect at a high level without having to open the Description; The name should be concise but detailed. Bug Names should be written in the following format: Screen > Brief Summary > Platform
    a. Ex: Assign Names Summary Page – Guest names displaying in dropdown incorrectly – iOS
2. **Bug State:** The following Defect States exist in JIRA: TO Do, Analysis, Ready for Dev, In-Dev, Ready for Testing, Test, Ready for Accepted, Blocked, and Done
3. **Description:** List the steps needed to re-create the issue, including how and what was found, the Expected vs. Actual Result and test data utilized (see below for further detail)
4. **Assignee:** The Scrum Master who will assign the bug to one of the developers on their team for resolution (see Appendix for list of Scrum Masters). Note: bugs cannot be assigned directly to Developers
    a. If it is unclear which Scrum Team to assign the bug to, consider whether the issue is occurring across all front-end platforms or if it is specific to a platform (I.e. Android vs. iOS vs. Web). If the issue is occurring across platforms, the backend might be the route cause
5. **Component/s:** The team that the bug is assigned to for resolution (I.e. Android Team, Commerce Team, ICE Web Team, Integration Architecture Team, Interact Team, iOS Team, Team Member Tools Team, etc.)
6. **Environment:** The environment that the issue was found in (I.e. QA, UAT, STG, Hagrid)
7. **Severity:** Ranked from Sev1-Sev5 based on business impact
8. **Creation Date:** The date the bug was created (this is automatically populated within JIRA)

Populating the Defect Description

When populating the defect description, it is important to provide enough detail for the Scrum Masters and developers to be able to decipher what the issue is as well as to recreate the issue at hand. The following information needs to be included:

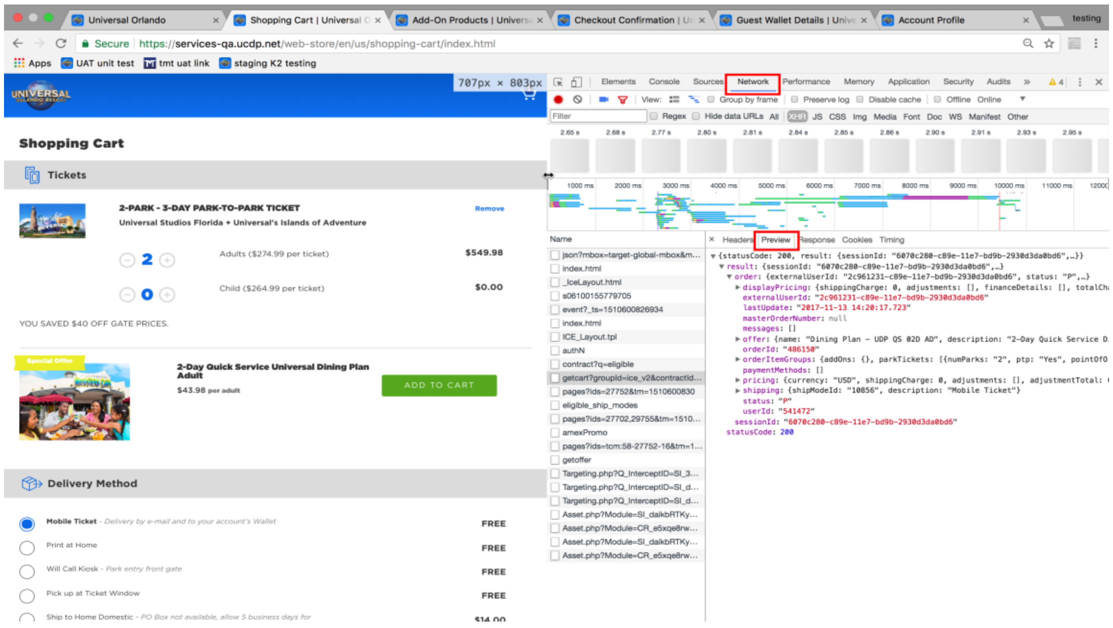
1. Steps to Recreate: Detailed instructions that allow the development teams to reproduce the issues on their own. This includes any account credentials, product types, order confirmation number, credit cards utilized, device / browser the issue was found in, etc.
2. Expected Result vs. Actual Result: The Expected Result details the requirement or exit criteria that is not being met. The Actual Result details how the functionality is inaccurately working in the environment

3. Miscellaneous: Any additional information that can be given to the developer as part of the triage process should be included. Even if it doesn't seem relevant, any additional information may assist in getting to a resolution

Dev Tools – Web Browser

When executing web test scenarios, Developer Tools can be utilized as an asset to gain insight into the specific calls taking place. How to access Dev Tools differs across browsers, but can usually be found in the Tool Bar menu. The information contained within these responses can be added to defect descriptions for an additional level of detail to aid developers with defect triage as the request and response provides them with the calls between the front-end and back end systems.

To view each service response throughout the session, navigate to the **Network > Preview** function.



https://services-qa.ucdp.net/web-store/en/us/shopping-cart/index.html

Shopping Cart

Tickets

2-PARK - 3-DAY PARK-TO-PARK TICKET
Universal Studios Florida + Universal's Islands of Adventure

2	Adults ($274.99 per ticket)	$549.98
0	Child ($264.99 per ticket)	$0.00

YOU SAVED $40 OFF GATE PRICES.

2-Day Quick Service Universal Dining Plan Adult
$43.98 per adult

Delivery Method

Mobile Ticket - Delivery by e-mail and to your account's Wallet	FREE
Print at Home	FREE
Will Call Kiosk - Park entry front gate	FREE
Pick up at Ticket Window	FREE
Ship to Home Domestic - PO Box not available, allow 5 business days for	$14.00

```
json?mbox=target-global-mbox&m…
index.html
_IceLayout.html
s06100155779705
event?_ts=1510600826934
index.html
ICE_Layout.tpl
authN
contract?q=eligible
getcart?groupId=ice_v2&contractId…
pages?ids=27752&tm=1510600830
eligible_ship_modes
pages?ids=27702,29755&tm=1510…
amexPromo
pages?ids=tcm:58-27752-16&tm=1…
getoffer
Targeting.php?Q_InterceptID=SI_3…
Targeting.php?Q_InterceptID=SI_d…
Targeting.php?Q_InterceptID=SI_d…
Asset.php?Module=SI_daikbRTKy…
Asset.php?Module=CR_e5xqe8rw…
Asset.php?Module=SI_daikbRTKy…
Asset.php?Module=CR_e5xqe8rw…
```

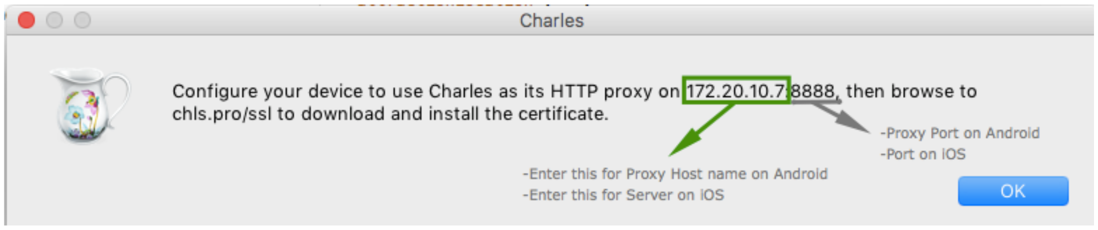
```
▼{statusCode: 200, result: {sessionId: "6070c280-c89e-11e7-bd9b-2930d3da0bd6",…}}
  ▼result: {sessionId: "6070c280-c89e-11e7-bd9b-2930d3da0bd6",…}
    ▼order: {externalUserId: "2c961231-c89e-11e7-bd9b-2930d3da0bd6", status: "P",…}
      ▶displayPricing: {shippingCharge: 0, adjustments: [], financeDetails: [], totalCha
       externalUserId: "2c961231-c89e-11e7-bd9b-2930d3da0bd6"
       lastUpdate: "2017-11-13 14:20:17.723"
       masterOrderNumber: null
       messages: []
      ▶offer: {name: "Dining Plan - UDP QS 02D AD", description: "2-Day Quick Service D
       orderId: "486150"
      ▶orderItemGroups: {addOns: {}, parkTickets: [{numParks: "2", ptp: "Yes", pointOfO
       paymentMethods: []
      ▶pricing: {currency: "USD", shippingCharge: 0, adjustments: [], adjustmentTotal: 
      ▶shipping: {shipModeId: "10856", description: "Mobile Ticket"}
       status: "P"
       userId: "541472"
    sessionId: "6070c280-c89e-11e7-bd9b-2930d3da0bd6"
  statusCode: 200
```

Charles Proxy

When logging defects it is important to provide the development teams with all information possible to aid in the triage process. A Charles Proxy is a way of capturing all of the service requests and responses while testing on a front-end platform. This allows for a detailed account of exactly how the defect was produced. Inputting Charles traces or specific error responses into defect attachments is expected when raising a defect, especially found via front-end test execution. To set up Charles on your laptop or desktop:

1. Download the free Charles application onto your computer from: https://www.charlesproxy.com/
2. Connect both your computer and front-end platform (I.e. Mobile device) to the same Wi-Fi network; we have found that mobile hotspots tend to work best

3. Launch Charles and navigate to *Help > SSL Proxying > Install Charles Root Certificate on a Mobile Device or Remote Browser*. A pop-up message below will appear, providing information that will need to be utilized



Configure your device to use Charles as its HTTP proxy on 172.20.10.7:8888, then browse to chls.pro/ssl to download and install the certificate.

Charles
-Proxy Port on Android
-Port on iOS
-Enter this for Proxy Host name on Android
-Enter this for Server on iOS
OK

4. **Enter the Proxy Host/Server and Proxy Port shown above into the Wi-Fi settings on your mobile device.** To locate these fields, see below:
   - **Android:** Press down firmly on the Wi-Fi network you are currently connected to and Manage Network Settings. Hit Save after entering info
   - **iOS:** Tap the info icon on your connected Wi-Fi. Scroll down to HTTP Proxy and select Manual
5. Go to www.chls.pro/ssl on your mobile device to download/install the certificate required
6. IOS only: Navigate to *Settings > General > About > Certificate Trust Settings* and enable full trust for the Charles root certificate
7. The Charles application should now track the actions performed on your mobile device

After the trace has been captured, detailing out the request and response of the issue at hand, upload the trace to the Attachments section of the defect logged.

Defect Life Cycle

There is a common path that all defects will follow being raised by a member of the QA Test team. The lifecycle of a defect is an important process to understand as it drives the progression of the solution and based on severity, must be resolved for User Story acceptance.

Defect States

Defect Severities

Functional defect severities are determined based off business impact and are reviewed frequently with the Product Owners / Analysts. User Stories cannot be sent for Acceptance with open Sev1 or Sev2 defects nor will the Release at hand go to Production. Lower severity defects are reviewed with the business throughout the QA, Regression and Integration Test Phases to ensure the business accepts the issue for Go-Live.

Severity Level	**Description**	**Remediation**	**Example**	**Turnaround Time**
1	Severe Business Impact: System down with no manual work around available	Escalate for immediate resolution. Resolution is required for successful completion of testing cycle	Critical failure to application availability, results, functionality, performance or usability. Unable to continue testing	1 business day
2	Significant Business Impact: System down but with manual work around possible	Escalate for immediate resolution and continue testing with workaround. Resolution is required for	Critical Business component unavailable or functionally incorrect. Workaround is available	Up to 2 business days

		successful completion of testing cycle	for testing to continue; however, this failure must be corrected prior to close out of the test cycle	
3	Minor Business Impact: Minor loss of service; the impact is restricted to few accounts. A work around is available	Log for resolution; continue testing available components. Resolution may be required for successful completion of testing cycle	Non-critical component unavailable or functionally incorrect; incorrect calculation results in functionally critical key fields/dates and workaround is available, even in Production	Up to 3 business days
4	Cosmetic Request: No loss of service; issues include incorrect message wording, formatting, or minor system errors	Log for resolution, continue testing. Resolution is not required for successful completion of testing cycle	Usability errors; screen or report errors that do not materially affect quality and correctness of function, intended use or results	Up to 5 business days
5	Enhancement Request: Individual component working as designed but enhancement requested	Log for resolution, continue testing. Resolution is not required for successful completion of testing cycle. This can be added into the next release	New functionality not critical to business operations. Or, could be a missing field on a screen or report	N/A

Non-functional, mostly visual, defect severities are determined based off business impact and are reviewed frequently with the Product Owners / Analysts a well.

Severity Level	**Description**	**Remediation**	**Example**	**Turnaround Time**
1	N/A	N/A	N/A	N/A
2	Significant Visual Impact: Visual defects that either (1) Have significant impact to the Guest experience, (2) Would be noticeably incorrect to the Guest, (3) Are detrimental to the brand, or (3) Are a Legal or licensed partner contractual issue	Escalate for immediate resolution and continue testing with workaround. Resolution is required for successful completion of testing cycle	Video or image not loading correctly; Misaligned dropdowns keep guest from accessing other functionality; Incorrect copy or image prevents Legal/LP approval; Significant font inconsistency, spacing issues or text wrapping; AP colors all the same on product cards; Form fields misaligned. Misaligned CTAs, chevrons and links; Page "jumps" on refresh	Up to 2 business days
3	Minor Visual Impact: Visual defects that (1) Have minimal impact to the Guest experience, or (2) Would not be noticeably incorrect to the Guest but would be noticeably incorrect to the business	Log for resolution; continue testing available components. Resolution may be required for successful completion of testing cycle	Incorrect button or promo color; Incorrect or inconsistent copy with no legal or LP impact	Up to 3 business days

4	Cosmetic Request: Visual defects (i.e. comp deviations) that have no impact to the Guest experience or the business, but would be noticeably incorrect to the Creative team	Log for resolution, continue testing. Resolution is not required for successful completion of testing cycle	Small spacing or padding inconsistencies as determined by creative team; Minor font weight/color/size inconsistencies as determined by creative team	Up to 5 business days

Workstreams

At Universal, there are several guest facing non-guest facing systems, platforms and applications that make up the ecosystem for DEP projects. The QE Test team is responsible for testing each system, platform and application to ensure a streamlined guest experience is provided for implementation.

Guest Facing

- **ACP**: ACP is the technology that manages and controls access to the parks as well as redemption points (I.e. Virtual queue, entering rides / park, redeem entitlements at tap points). The technology integrates with many of the other programs implemented in Project 406
- **Android**: Front end Android UO app development
- **Guest Profile & Account (GPA)**: Guests can create and edit their account, (I.e. Contact information, preferences, addresses, preferences, manage a travel party). All information on this guest facing function is saved in MDM AE
- **Interact/Personalization**: Leverages attributes known about a guest to select best product & content offers to be presented at strategic points throughout the front-facing applications
- **iOS**: Front end iOS UO app development
- **UO.COM**: is the current front end Production website for all products/experiences (I.e. Hotels) outside of the Web ICE (Commerce)
- **Payments/eFolio**: Payment processing, credit card tokenization and payment compliance, saving cards to wallet, tap-to-pay (utilized at the park with TapuTapu bands)
- **Team Member Tools (TMT)**: Application for UO team members to better assist guests (I.e. Search guest, add/manage entitlements & timeline)
- **ICE Web**: The Integrated Commerce Experience on Web (Tickets, UEPs, Extras) is front end development for the UO website

Non-Guest Facing

- **ACI**: Third party that processes payments during an ICE checkout. Whenever a purchase is completed or a credit card is added, all related information is passed to ACI for validation
- **Accertify**: Third party that validates payments for fraud. To trigger fraud, utilize the following phone number: (407)-222-2222

- **Business Intelligence & Reporting**: Gathers data and information on P406 (I.e. Dashboards, tags, data models)
- **Cloudant:** Cloud data storage and management system. This system interacts with both MDM AE / CE and the front-end to provide information quickly to other integrated systems. The information is housed in indexes, which allow for faster search and retrieval of information
- **Data/Architecture**: Design and manage architecture and data systems for P406
- **Microservices (Integration Architecture)**: Middleware that integrates the backend and front-end systems. Microservices takes information from the sender application and batches it in a way for the recipient application to read. These services are what allows our different systems to talk to each other
- **WebSphere Commerce**: An IBM application that allows for the Integrated Commerce Flow Interface. This application serves as a virtual store that the users interact with through the front-end ICE interface and mobile platforms
- **Master Data Management (Advanced Edition) (MDM AE):** Maintains and develops a relational database to story, analyze, and retrieve data about the customer (I.e. Guest Account & Profile)
- **Master Data Management (Collaborative Edition) (MDM CE):** Maintains and develops a database to store, analyze, and retrieve data about the product catalog (contains attributes for all product SKUs)
- **Security:** Authenticates and authorizes use of DEP service and entitlements (I.e. OAuth, Webseal and ISAM)

Types of Tests

Front End Validation

- Goal: Prove that the UX/UI displays the correct information or desired outcome
- Front end validations are performed by following an end user's path through the front-end platform (I.e. Mobile app, website, TMT) to complete a specific action (business processes and customer experiences). By achieving the desired outcome, front-end validations are proof that all data is flowing correctly through the involved back ends

Back End Validation

- Goal: Prove that the APIs, services and systems are transferring and utilizing the correct data
- Back end validations are performed by entering inputs on the front-end platforms and validating the accurate flow of data with the respective back end teams. Utilizing readily available tools such as Charles Proxy and Dev Tools, the testing team can view the request and response of each input. There are some backend systems that the QA Test team does not have access to for validation, however we can work with specific resources to obtain the necessary information (see the Resources section of the Appendix)

Inventory Testing

Within the DEP solution, there are subsets of inventory controlled products. The business must ensure that fulfillment of these products is being handled in such a way that the respective back end fulfillment system is updated in real time.

Hold Inventory

The inventory hold time is the amount of time a customer can keep items in their cart idle before the products are removed and made available for another customer to purchase. It is important to note that if a guest is currently holding the last piece of inventory for a specific item, no other customer will be able to purchase that item. Inventory hold times are configurable and different across inventory systems. Normal inventory hold times for Galaxy products are set to 15 minutes. During tight testing timelines, the hold time can be manipulated for efficient testing (I.e. One minute). The hold time configurations are configured by the Commerce team (David Walker or Matthew Voss).

Note: If a guest makes changes to items in their cart or adds any new products to their cart during the hold period, the inventory hold restarts. The customer is provided more time with that inventory because they are actively changing what they are trying to purchase.

Example Test Scenario: *Validate that inventory is held upon adding BMG to cart - iOS.* In this scenario, we are validating that when Guest1 has a specific BMG product in their cart and the hold has not expired yet, Guest2 cannot have access to Guest1's specific product. To test this, we would utilize two different devices side by side (I.e. Two mobile devices or one mobile device with the website). To make this a simple test, the BMG inventory would need to be configured to one available inventory so that we know exactly which product the two users are attempting to purchase. The test execution steps would be:

1. Guest1 using Device1 adds the BMG product to cart
2. Guest2 using Device2 attempts to add that same BMG product to cart while Device1's hold is still valid
3. The expected result is that Guest2 receives an error or sees the item as SOLD OUT because Guest1 still has that inventory held

Check Inventory Right Before Payment

This inventory check is utilized when two guests are purchasing the last inventory of a product at around the same time. We do not want Guest2 to get arrive at the checkout screen with the inventory Guest1 has just purchased nor to be able to complete the purchase. Specific error messages have been identified for these scenarios. In specific cases, a guest will be taken back to the beginning of the shopping flow if the inventory they are requesting is unavailable.

Note: These scenarios will be like Hold Inventory, except this inventory check is being validated on the checkout screen. In these scenarios inventory is automatically rechecked and given to the user if there is inventory still available. If there is only one inventory left and their hold expires, that item is now available for other guests.

Example Test Scenario: *Validate if ticket inventory is no longer available, user is returned to the cart page.* In this scenario, we are making sure that while Guest1 is on the checkout page with a specific ticket SKU their inventory hold has expired, Guest2 can have access to Guest1's specific product. To test this, we need have two different devices (I.e. Two mobile devices or one mobile device with the website). To make this a simple test, ticket inventory would be configured to one available inventory so that we know exactly which item the two users are trying to purchase. The test execution steps would be:

1. Guest1 using Device1 adds the product to cart and navigates to checkout, but does not complete the purchase. Their inventory hold then expires
2. Guest2 using Device2 adds that same product to cart while Device1's hold is now expired
3. The expected result is that Guest2 can purchase the item Guest1 no longer has a hold for

Commit Inventory

Upon checkout, a guest should have validation that the product(s) just purchased are in their possession. Even if processing issues are encountered, the guest will still have ownership of that inventory. To test these scenarios, we need to verify in backend systems that the inventory is indeed being reduced once a customer has purchased it. Inventory is committed in different locations for different types of products. If this is functions incorrectly, multiple customers could be under the impression that they have the inventory, but instead only some of them are truly getting the "committed" inventory to their wallets.
Note: The requirements for Commit Inventory have changed in the past few months. Inventory used to be committed differently depending on which delivery method was selected. For example, Ship to Home Domestic inventory was committed when the tickets were physically shipped from the post office. This massive time difference between purchase and commit was a problem. This requirement has changed to **committing the inventory right after purchase for all products and all delivery methods.**

Example Test Scenario: *Validate the selected UEP inventory is committed after successful payment authorization - ship to home (domestic).* In this scenario, the tester would complete a purchase with the specified delivery method. An email will need to be sent to the respective inventory system contact (in this case Galaxy) to validate the inventory has successfully been committed. The following needs to be included in the email: Account information, timestamp of when the item was purchased, specific product purchased, and order number.

Inventory Systems by Product

Inventory Management System	Inventory Controlled Products
Galaxy	▪ Universal Express Passes ▪ Annual Passes ▪ Seasonal Tickets ▪ Cabana Experience

	▪ Premium Seating ▪ Dining Plans ▪ Souvenir Cups ▪ Ticket + UEP ▪ VIP Experience ▪ Coca Cola Souvenir Cup ▪ Dining Plan Quick Service ▪ Dining Plan Quick Service + Souvenir Cup ▪ City Walk Meal and More ▪ City Walk Party Pass and More ▪ City Walk Meal and Mini-Golf ▪ City Walk Meal and Movie
NexTable	▪ Dining Experiences ▪ The Grinchmas & Friends Character Breakfast ▪ Universal Cinematic Spectacular Dining Experience ▪ Universal Superstar Christmas Breakfast ▪ Cabana Experience ▪ Premium Seating ▪ HHN ScareActor Dining
Shubert	▪ Blue Man Group Ticket Products ▪ Blue Man Group Show VIP Experience ▪ Bundle: Ticket + BMG

*Note: Cabana/PS/Dining are included in both Galaxy and NexTable because both inventory systems are utilized in the hold/commit inventory process
*Note: Inventory levels for each of these products are defined in the respective back end system. Updates to inventory are made in real time via Microservices that are integrated with the eCommerce capabilities of the Universal Orlando website and native mobile applications

Cross Browser Testing

Having a streamlined guest experience is a major part of DEP. With that comes the need for a consistent guest account across all front-end platforms (Android, iOS, TMT and Web). It is expected that a registered guest can make changes to their accounts, maintain items in their cart, maintain travel party members, as well as retain all previous purchases. No matter which platform a registered guest accesses their account from, they should have a consistent experience. Registered guests should be able to log into their Universal Account on any platform as well as see items added in cart, travel party members, and any previously purchased entitlements.

This testing traditionally executed during the Integration Test phase and includes validation across each platform to for the following scenarios:

- Guest Account changes (edit Contact, Personal, Address, Password, etc.)
- Add to Cart

- Products only available on Web can be viewed in Cart and purchased on both Mobile platforms
- Cart Merge

Accertify Testing

During the Integration and Regression Test Phases, it is important to validate that our fraud checks are functioning appropriately during completing a purchase. Accertify checks for fraud by utilizing algorithms based on the guest, credit card and address information.

To simulate fraud in our test environments, we utilize the following phone number upon checkout: 407-222-2222. When a tester utilizes this phone number while completing a purchase, it will appear as if the order was successfully processed on the UI (we still receive the confirmation number and success message), however this is not the case on the back-end. As the order is being passed to Galaxy to create the VisualID (VID) Accertify will flag the order for fraud, the ticket will NOT be issued and the order will need to be manually verified by IT Security. For testing purposes, we send over a list of order confirmation numbers that we have triggered fraud for to Brian Arnone and ask him to review / "release" the orders. The releasing of these orders will allow them to pass the fraud validation and continue to Galaxy for VID generation.

There are four different "resolutions" that Brian can initiate:
1. UO - Instant Trace: Order is instantly released from review and traced
2. UO - Instant Accept: Order is instantly released from review and accepted
3. UO – Trace: Order has a delayed release from review and trace
4. UO – Accept: Order has a delayed release from review and is accepted

It is important to point out that from a front-end test perspective, there is no difference between these four resolutions. Receiving a QR code in the Guest Wallet signifies the order was successfully released and accepted, no matter which resolution Brian chooses.

To execute Accertify test scenarios:

1. Complete a ticket purchase, with the Mobile delivery method and 407-222-2222 as the phone number during checkout
2. Take note of the UO confirmation number
3. Navigate to the Guest Wallet and validate that there is NO QR code generated (meaning the order was caught up in fraud review and did not make its way to Galaxy)
4. Email the UO confirmation number to Brian Arnone (brian.arnone@universalorlando.com) and ask him to "release" the order
5. Receive confirmation from Brian that the "order has been resolved" (meaning he released it)
6. Navigate to the Guest Wallet and validate there IS a QR code generated (meaning the order passed the fraud check and was sent to Galaxy for VID creation)

Appendix

Test Execution Links and Logins

ACP Terminal Logins

- Front Gate: Username – frontgatemanageruser / Password – passw0rd
- Band Activation: Username – superuseruser / Password – passw0rd
- Dispatch: Username – attractionsmanageruser / Password – passw0rd
- Return: Username – attractionsmanageruser / Password – passw0rd

Accesso Ride IDs

Ride Name	Accesso Ride ID
Honu	{ "10015", new { AttractionId = new Guid("95a7d631-d1db-41b6-9696-3a0d4835a032"), AttractionName = "Honu" }
Ika Moana	{ "10016", new { AttractionId = new Guid("f15ebd0c-8097-4d76-bb6c-18c9550f0d8c"), AttractionName = "Ika Moana" }
Kala & Tai Nui	{ "10017", new { AttractionId = new Guid("e15cfb4a-51d0-4af9-98c6-04e306fb4e58"), AttractionName = "Kala & Tai Nui" }
Ko'okiri	{ "10018", new { AttractionId = new Guid("5b309bb3-5cdf-4368-83f4-090461c36828"), AttractionName = "Ko'okiri" }
Krakatau	{ "10019", new { AttractionId = new Guid("5b43a70a-7781-4578-ade2-4f5ac7f1550f"), AttractionName = "Krakatau" }
Maku	{ "10020", new { AttractionId = new Guid("19c8b57a-033d-4b0e-9724-bcfcdbe03924"), AttractionName = "Maku" }
Ohno	{ "10021", new { AttractionId = new Guid("4d333ed5-8bf5-4ad2-98cc-8881a1a0007f"), AttractionName = "Ohno" }
Ohyah	{ "10022", new { AttractionId = new Guid("441d4a93-2195-46b9-a542-764858db83b1"), AttractionName = "Ohyah" }
Puihi	{ "10023", new { AttractionId = new Guid("d1b8e968-ebc4-4882-8d78-c939b69b0340"), AttractionName = "Puihi" }
Punga Racers	{ "10024", new { AttractionId = new Guid("cbd8dad6-759e-4fea-977c-f75b23d3ae94"), AttractionName = "Punga Racers" }
Taniwha Tubes	{ "10025", new { AttractionId = new Guid("39dd62ac-30d4-4a72-a3c1-06cc26b096ba"), AttractionName = "Taniwha Tubes" }

Hockey App Access (Mobile)

Mobile builds for test execution are delivered via Hockey App. Within the App there are sections for each build release as well as environments. To acquire access for Hockey App, email the following Universal Mobile team members:

Platform	Resource	Email
Android	Matthew Velie	matthew.velie@universalorlando.com
iOS	Matthew Velie	matthew.velie@universalorlando.com

Nextable DEV Logins

RestaurantID	Restaurant Name	Username	Password
T1H391X8B2	Cafe La Bamba Development (Superstar Character Breakfast)	uolabambadev	labamba!
T407W27FY1	Lombard's Development (Cinematic Dining Experience)	uolombardsdev	lombards!
Q06I8X447	HHN Scareactor	uohhndev	uohhndev!
BAML0O2HT	Grinchmas Development	grinchmasdev	grinchmas!
TRA99PWJZ	Volcano Bay Development	uodevelopment	development!
E4PP6ZX54	Rainforest Village Development	uorainforestvillagedev	rainforestvillage!
3T5L552HP	River Village Development	uorivervillagedev	rivervillage!
ISGBT8UGL	Wave Village Development	uowavevillagedev	wavevillage!

Note: The NexTable User Guide can be found in the following Wiki: https://urldefense.proofpoint.com/v2/url?u=https-3A__wiki.inbcu.com_display_UPRAI_Cabana-2Band-2BPremium-2BSeat-2BInventory&d=DwIFAg&c=jf_iaSHvJObTbx-siA1ZOg&r=rmdPSGxeUV6KYlStxdjbO8Ayt9ECVS0-nHfMFcXd76A&m=7y989aSrUNqfkaSvXFPlGAxUPoNneK5Ll3bE8Ns43ZI&s=as_KwIWrZxiGEvm-IvFaPl_nhS9SmXsngmhRP1gQ2UY&e=

FreedomPOS (FPOS)

How to Launch:

- Ensure the PCI Port is activated and connected to the FPOS terminal (blue cord)
- Launch FPOS from the Terminal Desktop

How to Identify the Current Environment:

- Navigate to Windows Services (Start > Services)
- Open File Explorer and navigate to C:\Users\fpos\ms-pos\config\env
- Open the pos_uat.js file (right click and "edit with notepad++")
- Locate the line that notates msEndpoints: paymentsUrl:
- Locate the environment in the service URL (I.e. https://pay-**qa**.use.ucdp.net OR https://pay-**uat**.use.ucdp.net)

How to Change Environments:

- Navigate to Windows Services (Start > Services)
- Open File Explorer and navigate to C:\Users\fpos\ms-pos\config\env
- Open the pos_uat.js file (right click and "edit with notepad++")
- Locate the line that notates msEndpoints: paymentsUrl:
- Locate the environment in the service URL and update the to the desired environment (I.e. https://pay-**qa**.use.ucdp.net OR https://pay-**uat**.use.ucdp.net)
- File > Save
- Clear cache for "ms-pos" for the environment change to save

How to Clear Cache:

- Navigate to Windows Services (Start > Services)
- Locate 'ms-pos' and Restart the service

FPOS Store Login Credentials:

- Operator Number: 206497989

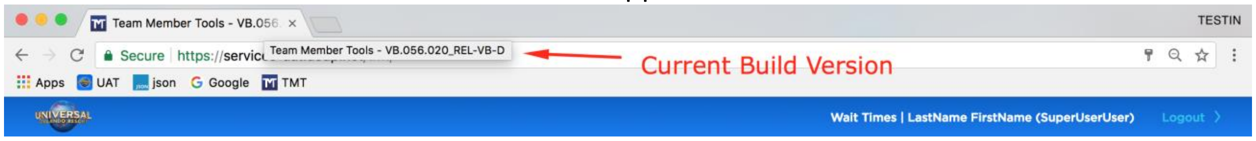
TMT Login and Links

Username: superuseruser
Password: passw0rd

Environment	Link
QA	https://services-qa.ucdp.net/tmt/
UAT	https://services-uat.ucdp.net/tmt/
STG	https://services-stg.ucdp.net/tmt/
STG02	https://services-stg02.ucdp.net/tmt/

How to Identify the Current Build Number:

- The build version can be located in the application browser tab:


Team Member Tools - VB.056
TESTIN
Secure https://servic
Team Member Tools - VB.056.020_REL-VB-D
Current Build Version
Apps
UAT
json
Google
TMT
UNIVERSAL
Wait Times | LastName FirstName (SuperUserUser)
Logout

Role Based Permissions (Users for Test Execution):

The following users exist in the UAT (Fiona) environment currently. Refer to US25571 for Role Based Permissions requirements. Note: The Passwords have been reset in QA. To set new users / passwords, contact Preston Wade (preston.wade@universalorlando.com)

User	Password	Bound Permissions
9999010	Lakernation123	"PremiumServicesAttendantsVL" has the inability to transfer an entitlement used or un-used
9999012	Lakernation123	"PremiumServicesLeadsCabanaConcierge" has the ability to transfer a used entitlement
9999016	Lakernation123	"ConciergeAttendants" has the inability to transfer a used entitlement, but transfer an un-used entitlement

Contacts for Configuring / Switching Hardware Over in the B17 Lab

Hardware / Config.	Contact
ACP Terminals	Jason Holmes: jason.holmes@universalorlando.com Jesse Martin: jesse.martin@universalorlando.com Miguel Montijo: miguel.montijo@universalorland.com
Galaxy Terminals	Brandon Helgeson: brandon.helgeson@universalorlando.com Hervence Gourdet: hervencegourdet@universalorlando.com
TMT Tablets	Sebastian Bylinksi: sebastian.bylinksi@universalorlando.com Jeanna Anderson: jeanna.anderson@universalorlando.com
Unicorn	Briana.peach@univeresalorlando.com Mital Shah: mital.shah@univeraslorlando.com

Web ICE Breakpoints, Browsers and Links

Breakpoints:

- Desktop: 1024

- Tablet: 768
- Mobile: 320

Browsers:

- Primary browser for QA Test Phase testing: Chrome
- Browsers for the Integration and Regression Test Phases: Chrome, Edge, Firefox, IE11, Safari

Users can navigate across the Web ICE site by clicking below direct links.

Environment	Link
QA	https://464.universalorlando.com/web-store/en/us/ park-tickets
UAT	https://131.universalorlando.com/web-store/en/us/park-tickets
STG	https://secure.universalorlando.com/web-store/en/us/park-tickets/index.html
STG02	https://stage02.universalorlando.com/web-store/en/us/park-tickets
PROD	https://www.universalorlando.com/web/en/us

Mobile Devices and Operating Systems:

Click below to get all detail on confluence page.
https://wiki.inbcu.com/display/UPRMA/Test+Devices

Test Credit Cards

When to Utilize	Credit Card
QA, Integration, Regression, Performance, E2E, UAT Test Phases	3782 8224 6310 005 (American Express)
	3714 4963 5398 431 (American Express)
	3787 3449 3671 000 (American Express Corporate)
	5610 5910 8101 8250 (Australian BankCard)
	6011 1111 1111 1117 (Discover)
	6011 0009 9012 9424 (Discover)
	5555 5555 5555 4444 (MasterCard)
	5105 1051 0510 5100 (MasterCard)
	4111 1111 1111 1111 (Visa)
	4012 8888 8888 1881 (Visa)
	4222 2222 2222 2 (Visa)
Bypass (When ACI is Down)	8080 0000 0000 0022 (Mobile)
	4012 8888 8888 1881 (Web ICE)
Decline Credit card information for Negative testing purpose	Discover - 6011000000001127
	Visa - 4005520000000343
	MC - 5471320000019944
	AMEX - 379605170000896

UPC Codes

The below list of UPC codes were created by the Product Owners (ICE) and loaded / configured in the test environments by the Galaxy / MDM CE (Product Planning) Team:

- 049000073461

- 049000133752
- 049000162738
- 049000171983

Inventory Controlled Products:

- Universal Express Pass (UEP)
- Blue Man Group (BMG)
- Dining Restaurants
- Cabanas
- Premium Seating
- VIP Experience
- HHN

Acronyms

Acronym	Definition
533	Project code for Volcano Bay
ACI	*Not an acronym; universal payments company with PCI
ACP	Access Control Program
API	Application Program Interface
BA	Business Analyst
BI	Business Intelligence
BI	Business Intelligence
BMG	Blue Man Group
DE	Defect
DMP	Data Management Platform
GD/GDC	Global Delivery/Global Delivery Centers
GPA	Guest Profile & Account
ICE	Integrated Commerce Experience
JSON	Javascript Object Notation
K2	Project code for UO website revamp
MDM	Master Data Management
MS	Microservices
OLTS	Online Ticketing Store
PA	Product Analyst
PAYG	Pay as you go
PCI (architecture)	Predictive Customer Intelligence
PCI (eFolio related)	Payment Card Industry
PI	Program Increment (synonym for “phase” or “release”)
PMO	Project Management Organization
PO	Product Owner

PRP	Password Reset Portal
QA	Quality Assurance
SEV	Severity (regarding defects)
TiA	Tickets in App
TMT	Team Member Tools
UAT	User Acceptance Testing
UI/UX	User Interface/User Experience
UO	Universal Orlando
UPR	Universal Parks and Resorts
US	User Story
VB	Volcano Bay
VL	Virtual Line
VQ	Virtual Queue
WCS	WebSphere Commerce Server
WnW	Wet n' Wild
UEP	Universal Express Pass