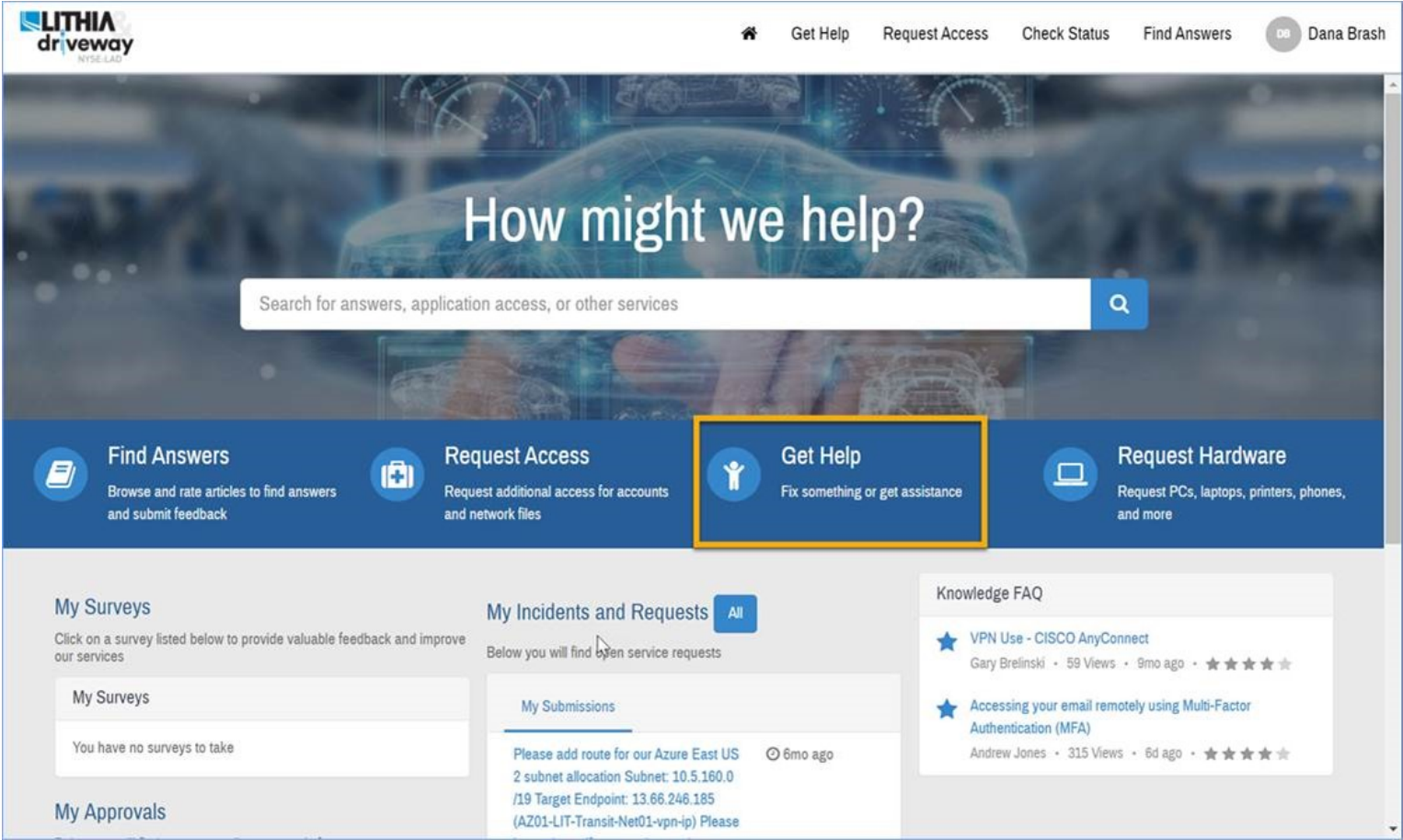


Can I close a work item if we determine the work is no longer required?

- If the PM, BA, and/or SDM confirm the work is no longer needed the work item can be closed. Case by case decision if we will replace the work item if it is in a sprint.

##Q: How do I engage the Cloud Platform Engineering (CPE) Team? A: The process to make requests to the Cloud Platform Engineering (CPE) team directly through the ServiceNow portal

- From the Service Now site, select “Get Help”



- Click on the “Cloud Engineering” tile:

Categories

- Home Office Support
 - Enterprise Architecture Review
 - Information Security
 - Project Support
 - SPE/Dashboard and Data Warehouse
- Technology Orders
- Workday
- Get Help**
 - Services
 - Request Access
 - Technician Catalog
 - Terminate User Access
 - Working At Home

Get Help

Fix something or get assistance

Get help with Driveway



Use this to get assistance with Driveway errors or issues

[View Details](#)

ServiceNow Team Intake

Get help from the ServiceNow Platform Team

[View Details](#)

Application and Website Issues

Get support for website and application problems

[View Details](#)

Application Installation

Get assistance with program installations

[View Details](#)

Cloud Engineering

Get help from the Cloud Engineering team

[View Details](#)

Computer Setup

Request assistance setting up your new computer/laptop and additional peripherals

[View Details](#)

Office 365 Password Reset

Self-service password reset for Office 365 services including Outlook.

Password Reset

Request a password reset for an application or service

Phone Support

Request support from our communications team

- Provide us some very basic information, description, and/or attachments and click submit

Cloud Engineering

Get help from the Cloud Engineering team

Get help from the Cloud Platform Engineering team for cloud application platform development, IaaS, PaaS and SaaS services including Virtual Machines, Data Services, App Services, Kubernetes, or other enterprise systems

* Who is the request for?

Dana Brash

Submit

5

Required information

Which application or service do you ne...

How can we help?

Please describe your request or issue?

1 * Which application or service do you need help with?

- ☐ Virtual machine (VM)
- ☐ Database
- ☐ Storage
- ☐ Data Factory
- ☐ AppService
- ☐ Network
- ☐ Kubernetes
- ☐ Enterprise Applications/SSO
- ☐ Data Analytics

2 * How can we help?

- ☐ Create resource
- ☐ Change resource
- ☐ Remove resource
- ☐ Connectivity
- ☐ Permissions
- ☐ Investigation
- ☐ Backup/Restore
- ☐ Other

3 * Please describe your request or issue?

4



Add attachments

What the CPE members see:

1. We get an Email:

New request RITM0149149 - Get help from the Cloud Engineering team



IT Service Desk <lithia@service-now.com>

To CloudEngineering



Reply

Reply All

You replied to this message on 11/17/2021 5:15 PM.

Cloud Platform Engineering,

This is a notification that a request was opened just now by Dana Brash. Please review the incident at your earliest convenience.

Request: [REQ0147664](#)

Number: [RITM0149149](#)

Short description: Get help from the Cloud Engineering team

Opened Date: 11/17/2021 05:13:39 PM PST

Requested by: Dana Brash

Requested for: Dana Brash

Requested for Office: ORLITMOT

Email script render error: email script [sc_req_item_approvers] does not exist

Category: IT Cloud Engineering

Sub-Category: Virtual machine (VM)

Tertiary Category: other

Thank you,

2. In ServiceNow we see a Ticket:

Service Management

DB Dana Brash REQ0147664

Filter navigator

Home

Service Desk - Incidents

IT Systems - ALL

IT Systems - DANA

IT Systems - ACTIVE

Change - Create New

Dana Change Requests

Service Desk - My Work

Incident - Assigned to me

My Requests Filtered

Knowledge - All

Self-Service - Requested Items

Requested Items

RITM0149149

Number: RITM0149149

Item: Cloud Engineering

Request: REQ0147664

Requested for: Dana Brash

Request Department Category: IT Cloud Engineering

Sub-Category: Virtual machine (VM)

Tertiary Category: other

Short description: Get help from the Cloud Engineering team

Opened: 11/17/2021 05:13:39 PM

Opened by: Dana Brash

State: Open

Stage: Request assigned

Assignment group: Cloud Platform Engineering

Assigned to:

Related to Covid-19?

Variables

* Who is the request for?

Dana Brash

* Which application or service do you need help with?

Virtual machine (VM)

Database

Storage

Data Factory

AppService

Network

Kubernetes

Enterprise Applications/SSO

Data Analytics

* How can we help?

Create resource

Change resource

Remove resource

Connectivity

##Cloud Platform Engineering ServiceNow Tickets: Process After Submitting Requests - Effective: 1/27/22

1. CPE team reviews the SNOW tickets daily during standup and determines if the request is a break fix or not.
2. For requests that do not have enough information for CPE team to triage during daily standup: -- 1. The Ticket Status will be changed to "Awaiting User Info" with feedback to the user in the ticket on what information is needed to triage the request.
3. For requests identified as a break fix: -- 1. Within 1 business day the CPE team will reach out to you regarding the break fix with any questions and keep you informed along the way until the issue is resolved. -- 2. CPE team prioritizes break fix requests over the items scheduled in the current Cloud Platform Engineering sprint and backlog. -- 3. Break fix requests are not entered into CPE Azure DevOps as a work item since they will be worked on right away.
4. For requests identified as not a break fix: -- 1. Within 1 business day you will get tagged in a new work item in our Cloud Platform Engineering Azure DevOps that is added to our backlog. -- 2. We will communicate with you in the Discussions section of the ADO ticket. You will get emails from the discussion from "Azure DevOps azuredevops@microsoft.com". Please be on the lookout for these emails. -- 3. We will provide you a link of a project spec doc (if required) to fill out in that work item and ask you any other questions needed to groom the ticket in the next scheduled CPE backlog grooming session. ----- A Project Spec Sheet will be required when the scope of the change is determined to include a significant effort and or change to the environment. ALL new to CPE resource types will require a Spec Sheet. -- 4. Once the request is groomed, the item will be pulled into a future sprint. Depending on the prioritization of the request, it may be the next upcoming sprint or several sprints later. ----- New requests that are classified for execution and tracking in ADO will be assigned story points, priority, and risk. This weighted calculation will determine how the work is scheduled and balanced against existing project work.

How to get an update on your ServiceNow (SNOW) or ADO ticket?

- For break fix work – Reach out to me (AliciaLyons@lithia.com) and I will coordinate with the team to get you an update. The user can also mention the engineer assigned to the ticket within SNOW.
- For non-break fix work – Please utilize the Discussion section in the CPE ADO Work Item to mention the engineer assigned to the Work Item in ADO.

What is considered a break fix?

- When an existing service or product stops functioning as expected or is functioning in a way other than intended.

- Break-fix priority is influenced by the scope of the issue (companywide, department wide, team wide, individual).

##Q: What do I do if I have an LDP Production Pipeline issue? (esply. with SelfHosted IR, restarting or recycling the Azure services etc) A: For PROD issues (esply. with SelfHosted IR, restarting or recycling the Azure services etc) , we have been asked to contact itsystems@lithia.com The team has On Call support and the requests will be picked up on incident priority.

All Pay; All Platforms; AllPay; APP-Dependency; Application-Performance; Azure; Azure-Environment-Setup; Bi-Weekly; Blocker-Payroll-Process; C#; Classic; Classic Bi-Weekly; CSS; DCH; Dev Config; DIP-Sheet-Report; Docs; e; Environment-Set-Up; Escalated; ETL process change; ETL-Blocked; ETL-Dependency; ETL-Documentation; ETL-POC; ETLsprint10; ETLsprint11; ETLsprint8; ETLsprint9; feedback; fly-by-report; Follow Up; Future; GIT; GL-Translator; Hardening; hotfix; HRMS-to-Workday-Change; JS; late commit; List of 21 Items; Macros/Workbooks; Marquam; Monitoring-and-Performance; MVP-Approved; Needs-Clarification; needs-dana; Needs-reqs; New-Change-Requirement; Nice; Onboarding; Pair-Programming; Parallel-Testing; PAYPLAN; Performance; PHASE 1; Possible-Resolution; Post-Workday-GoLive; PPE5/15-Day1-Issue; PPE5/15-Day2-Issue; Priority-6; Production Issue; QA-Complete; reflect-hub; Reports; Required-for-pay-plans; Review; Security; Security & Compliance; Seed Data; Service-Now; SmokeTest Bug; Source Control; SPIKE; SQL; SR1; SR1+; Standard Semi-Monthly; Standard Weekly; Store Rollout 1; Store Rollout 2; Store Rollout 3; Tech Debt; Testing; TMO; TPM; UAT-Deployed; unplanned work; Unplanned-Sprint-Item; v1.2.11; v1.2.13; v1.2.14; v1.2.15; v1.2.16; v1.2.17; v1.2.18; v1.2.19; VBA; Weekly; Workday; Workday Priority; Workday Regression Bug

As of 01/11/2021

Under Construction...

User Story and Bug workflow

The following workflow is for work items that are of type User Story or Bug. The emphasis here is on closing items within the boundaries of the sprint. By including these closed items in a Release Candidate at a later date, we have decoupled the work item's development workflow from the release process.

(It should also be noted that the effort required to take a Release Candidate through testing and ultimately to Production can be reflected in the Release work item's story points. This then contributes to more accurate velocities and aids the ability to plan future sprints.)

flowchart LR; New --> Ready --> Active <--> DevReview[Dev Review] --> Closed Active <--> Blocked

Status Descriptions - User Stories and Bugs

Status	Description
New	Created, but not yet ready to be worked on in a sprint.
Ready	Reviewed, refined, and pointed by the team. Ready for commitment.
Active	Assigned to a team member and actively being worked on.
Blocked	The work item is blocked or impeded and requires special attention.
Dev Review	The implementation is being reviewed by one or more team members.
Closed	Complete - No further work is required. (Add "Release Ready" tag.)

Release Candidate workflow

A Release Candidate is a work item that references all closed User Stories and Bugs since the prior release.

flowchart LR; Ready --> qReady[QA Ready] --> uReady[UAT Ready] --> pReady[Prod Ready]--> Closed

Status Descriptions - Release Candidates

Status	Description
Ready	The Release work item has been created and associated with all relevant work items.
QA Ready	The related build has been deployed to the QA environment for testing.
UAT Ready	The build has been deployed to the UAT environment for user acceptance testing.
Prod Ready	UAT is done and the release can be deployed into Production.
Closed	Production deployment is complete.

Sprint Status Appendix

New

- The work item has been opened but requires more information before it can be reviewed and scoped for LOE.

Ready

- The work item has a clear description of the work needed.
- The work item has clear acceptance criteria (if required).
- Both the description and acceptance criteria should be reviewed and accepted by
 - Technical Lead
 - BA
 - TPM
 - Business Team Representative (record approval in ADO)
- The work has an estimated score indicating the level of complexity.
- Engineers - have either added sub-tasks to the work or agreed that it does not require any.
 - Child task should be created titled "QA Notes" and all information/steps should be included there.
- Engineers - have added implementation notes or agreed that it does not require any
- Engineers and TPM - Identify the work risk level and determine if this requires a CAB invocation and scheduled release.
- The work item is ready to be committed to a sprint.

Active

- The developer is assigned and is actively being worked on by a developer.
- If the developer is no longer actively working on the item it should be moved back to Ready
- If the developer is blocked the work item should be moved to Blocked.
- If the work item requires another developer to review it should be moved to developer review and assigned to another developer to review the work item.
 - This includes Pull Requests, Developer QA, Technical Reviews
- If the work item does not require developer review, it can be moved merged into the 'Development' branch and deployed to Staging. The work item can be moved to QA Review.

Developer Review

- The developer's branch has been pushed to the repository and a pull request has been created for a developer to review it. The reviewer can either accept the changes or respond for further changes.
- Reviewer Checklist
 - An ADO work item has been associated with the PR
 - Link parent user story or bug (not tasks).
 - Run tests with 100% passed (Eventually when we have tests)
 - **Run the project successfully** (don't assume)
 - **The engineer has added QA notes**
 1. If story includes DB changes, attach SQL script(s) for QA to run for testing.
 2. If front-end, should be written in a way that could be passed onto the functional team for UAT.
 3. Time for this activity should be factored into effort estimates moving forward.
 4. Consider adding output of QA Notes into parent story instead of child task (but also include child task to track effort for it).
- Review PR code and check for the following
 - Large blocks of commented code
 - Large functions that can easily be refactored
 - Naming conventions make sense
 - Unused variables or functions
 - Adherence to (or violations of) [SOLID principles](#)
 - etc.. **###Release Ready - DEPRECATED**
 - The work item has been merged and deployed into the development environment
 - The development environment and all work items within it prove to be dev tested and stable
 - Communication with BA/TPM/SDM to coordinate a release to be cut and merged into the QA environment
 - Coordinate with the QAE (or second developer) and Review the [semantic version](#) that the release should be tagged before the release is cut.
 - If a major or minor version is declared, we need to communicate this to the larger team(s) for review of possible risks or the changes.

###QA Ready

- The release has been deployed to the test environment and is ready for regression testing.
- The associated work items are ready for internal QA by the TPM or BA.
- QA work should be tracked in designated User Story, not as task on development User Story.
 - This makes it easier to track QA that extends past the sprint (expected) and provides more accurate reporting around "carried over" items.
- When testing specific stories and bugs in the release, if the work passes acceptance criteria the tag "QA-Complete" may be added to the work item. For projects where the source control branch is coupled to the environment (Source/Environment-Coupled), the BAs/TPM will give the go ahead to merge the test branch into UAT. Otherwise, the release build can be deployed to UAT when all stories and bugs are tested, and when regression testing has passed.
 - *NOTE! - BAs and Developers need to coordinate right before we cut a release from development so that the code is "Frozen" as we merge it into the test branch. Please make sure we know what items are in that release.* Please refer to the [Environment Release Versions](#) (This is only for Source/Environment-Coupled projects.)

###UAT Ready

- When the Release has passed QA and is ready for UAT
- NOTE! (For Source/Environment-Coupled projects) The BA/TPMs will determine when the test branch should be merged into UAT and deployed to the UAT environment.
 - Developers should not be merging into UAT without permission from the BA/TPM/SDM
 - This deployment has a checklist which and should be followed when releasing updates into UAT
- Once deployed, BA/TPM will Assign the item to a representative of the functional team and if necessary, organize time for the functional team to have a session with the BA or Developer to walk through how to test and sign-off on the work.
- Once the functional team approves of all changes the functional team member assigned will note their approval in the comments of the item in ADO and change the state to "Prod Ready" and assigned to the technical lead. The BA/TPM will work with the functional team to schedule a time to deploy the changes to Production.

###Prod Ready

- When the release is ready to be deployed into production
- The Release Engineer will follow the [release checklist](#) prior to deploying to production
 - Please use [Semantic Versioning](#) for release tagging (ie v1.0.0 > 1.1.0) - NOTE! There could be more than one change which is why there needs to be release notes of what will be deployed so that the functional team can validate the changes in Production. The Release Engineer will verify that release notes are available in [Environment Release Versions](#) - Release notes will be shared with the team regarding what was released post deployment.

###Closed

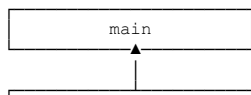
- The work item (user story, bug, or release) is completed. The "Release Ready" tag should be added so that the work item can be found easily and added to a release candidate.

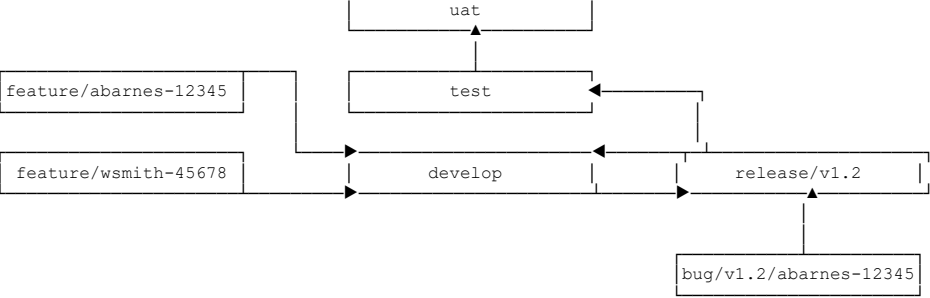
###Blocked

- The work item can't proceed because of an impediment external to the team.
- This status requires special attention and should be brought up in stand-up or as early as possible.

[Git Branch Organization](#) [How to name your branches](#) [Deploy/Develop into Test](#) [Notes about Merging Branches](#) [Additional Information & Resources](#)

Git Branch Organization





#How to name your branches

Type/name-number

Developer-Name

- first initial of first name and last name
- example: Andy Barnes = 'abarnes'

Type

- The type of work outlined in DevOps
 - feature (user story)
 - bug
 - hotfix
 - release

Item Number

- The item number associated in the DevOps sprint
- If there is not an item number associated, use a description key word ('demo', 'test')

Branch naming examples:

- feature/abarnes-1234
- bug/v1.2/wsmith-3456
- bug/abarnes-12345
- hotfix/v1.0/jadams-1232
- release/v1.2

#Deploy Develop into Test

Creating a release branch

Release branches are created from the develop branch. For example, say version 1.1.5 is the current production release and we have a big release coming up. The state of develop is ready for the “next release” and we have decided that this will become version 1.2 (rather than 1.1.6 or 2.0). So we branch off and give the release branch a name reflecting the new version number:

```
$ git checkout -b release/v1.2 develop
Switched to a new branch "release-1.2"
-- bump version in future (Currently working on that...)
$ git commit -a -m "Bumped version number to 1.2"
[release/v1.2 74d9424] Bumped version number to 1.2
1 files changed, 1 insertions(+), 1 deletions(-)
```

Edit Readme.md

Change the version number to the release version (1.2) and save the file.

Finishing a release branch

When the state of the release branch is ready to become a real release, some actions need to be carried out.

The newly created release branch should be merged into the test branch

```
$ git checkout test
Switched to branch 'test'
$ git merge --no-ff release-1.2
Merge made by recursive.
(Summary of changes)
$ git tag -a 1.2 -m "test version 1.2"
```

Deploy to the test environment

- Work with the QA team members to schedule a deployment time where it won't disrupt their current testing.
- Once deployed to the test environment, please publish the release notes and notify the QA team.

Semantic Versioning Protocol

See [Semantic Versioning](#).

The release is now done, and tagged for future reference.

#Deploying to the UAT Environment

Merge test into UAT

Merge the test branch into UAT and tag the associated version. If you don't know the current version use `git tags`. Please also tag the test version with the current test version.

```
$ git checkout uat
Switched to branch 'uat'
$ git merge --no-ff test
Merge made by recursive.
(Summary of changes)
$ git tag -a 1.2 -m "uat version 1.2"
```

Deploy to the uat environment

- Work with the QA team members and functional to schedule a deployment time where it won't disrupt their current testing.
- Once deployed to the uat environment, please publish the release notes and notify the QA team.

Notes about merging branches

- Be sure to pull the latest changes and merge them into your feature branch before pushing the branch and merging it into the parent branch.
 - This will avoid merge conflicts in the parent branch
- Please be sure that all your tests are passing before merging your code into the parent branch
 - This is to avoid broken builds
 - In the future, we will introduce test automation that will reject merging feature branches if tests do not pass.

Additional Information & Resources

For more information on tagging please see <https://git-scm.com/book/en/v2/Git-Basics-Tagging> For more information about semantic versioning please see <https://semver.org/>


Summary definition of work items

Type	Description
User Story	A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.
Research Spike	A user story for which the team cannot estimate the effort needed. In such a case, it is better to run time-boxed research, exploration to learn about the issue or the possible solutions. As a result of the spike, the team can break down the features into stories and estimate them.
BA Work	Defined work that contributes to planning through working with the product owner to define the vision for a product and purpose of the project
Child Task	More tactical definition around the breakdown of a parent work item. Tasks are chunked out to consumable, measurable work output

#User Stories

##Writing and evaluating user stories before sprint planning User stories describes an end state when a user is able to complete the task or achieve the goal. Consider the example user stories when writing up a user story. ##User Story Format

GOAL | What is the goal of the story? (CRUD)

 Unassigned

0 comments

Add tag

State	● New	Area	Advanced Incentive Program\AllPay Application
Reason	New	Iteration	Advanced Incentive Program

Description

- **What is the business problem or opportunity?**
 - **User Persona (The Who)**
 - **Requirement (The What)**
 - **Business Value (The Why)**
- **User Scenarios (How is the system used)**
- **Additional Information to support the story**
 - **Current State**
 - **Assumptions (must be validated)**
 - **Considerations**

Acceptance Criteria

- What does it mean for this story to be 'Accepted'

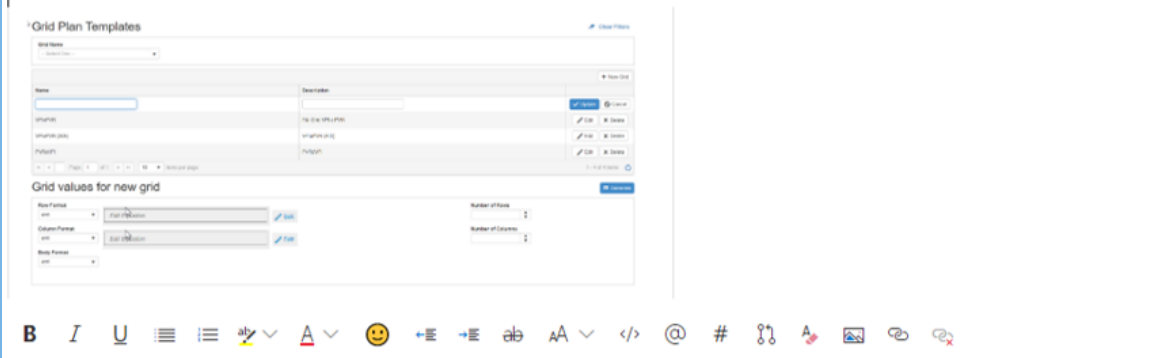
##Example Front End User Story

91577 Grid Plan | Update UI to use Equation Builder vs. Single Field

Description

As a Compensation Analyst

I want to use the equation builder to define grid plans so that my options for defining plans are not limited to a single field to build the formula.



Acceptance Criteria

- The ROW & Column data can be defined using the equation builder
 - Single field is no longer supported
- Selecting EDIT will trigger the Equation Builder app
- Data test tags are added for the Modal to support automation
- A feature flag is used to turn the feature on/off
 - It is expected the Single Field grid plan feature functions when the flag is ON

##Example Back End User Story

103811 Bulk Processing | BE | Evaluate core.t_InputSheetData for existing data

Unassigned

0 comments

Ready to Refine X +

State	New	Area	Advanced Incentive Program\AllPay Application
Reason	New	Iteration	Advanced Incentive Program

Description

As a Developer

I need to know if store input sheet data exists when inserting new records
So that store input sheet data is not overwritten.

Acceptance Criteria

- Does Store Input Sheet data exist? YES Then do nothing and skip to next
- Metrics are captured and returned to the front end
 - Return success
 - In the return object data - Indicate the total number of records evaluated
 - In the return object data - Indicate the total number of records that failed validation
 - For the selected Pay Group Region, Store Input Sheet data exists for a Company and PPE
 - In the return object data - Indicate the total number of records that were successfully processed
 - In the return object data - Provide an array of the rows that failed validation with the following information
 - Row number
 - Validation failure description (see validation rules and description document)
 - In the return object data - Indicate the total number of records that were ignored (due to missing earnings code and value).
 - Upon processing the respective report, if the process completely fails during evaluation please return a failure to the front end in which the FE should handle gracefully

Out of Scope:

- Rendering the results of the data load to the FE
- User personas (Required)
 - For Whom? If there are multiple end users, consider making multiple stories. Provide context and give an explanation of why. What is the goal of this story and the expectation of the result of it beyond the general persona need.
- Goals to achieve (Required)
 - What are the general goals that this story should achieve once the work is complete and working in production
- Current state vs Future State (Recommended)
 - What is the current "as-is" state of the work item before the desired work take effect. Explained what the future state will be after the desired work is completed. Be sure to not solution and instead, provide the future state from the users perspective.
- Assumptions (Optional)
 - There are times when the story writer assumes something that is not validated as fact. Assumptions can be made about the work but would require others to review, validate, or correct the assumption(s)
- Considerations (Recommended)
 - Does this story require documentations or additional closure requirements to call this "done" ?
 - Does this story require any follow up work or "technical Debt" that we need to identify to ensure things we are not able to get to are tracked?
- User Scenarios (Required)
 - Define user scenarios and expected outcome for each scenario
 - Both positive and negative user scenarios are included and expected outcome
 - If user scenarios aren't needed, please indicate that so the QA Rep can review and validate this.
- Acceptance Criteria (Required)
 - Acceptance criteria helps to reach an agreement between the team and clients regarding the product's functionality and completion of development. It needs to be clear as to what is expected in order to define the work as "done". Avoid using the description for embedding acceptance criteria. Acceptance Criteria should be easy to identify and read to know what is required.
 - Adds certainty to what the team is building
 - Must be testable with definite pass/fail outcomes
 - The criteria should be straightforward and compact
 - Easy to understand
 - Written from the actual user's perspective
 - Used to set boundaries and define the project's scope

#PAP meeting comments to be incorporated:

- What areas of the system may be impacted?

- Do we need diagrams or data flow to support this story? - Called out in BA Work times WIC
- Does this work item have any variation between environments?
- Proposed template to draw from

##Refining and evaluating user stories before sprint commitment Consider the following when reviewing and planning before commitment of this work into a future sprint.

- If there are conditions where there are multiple steps in the process completing the work in a story, call that out in the description to provide even greater clarity around dependencies or other conditions that may need to be factored in when planning out the work of the story. Write a story for each step in a larger process.
- Listen to feedback
 - Talk to your users and capture the problem or need in their words. No need to guess at stories when you can source them from your customers.
- Outline subtasks or tasks
 - Decide which specific steps need to be completed and who is responsible for each of them.
- Time
 - Since stories should be completable in one sprint, stories that might take weeks or months to complete should be broken up into smaller stories or should be considered their own feature or epic.

##Reviewing the output of the work conducted within the User story

- Dev Review Stage
 - User stories normally have a Dev Review process in which a fellow developer will review the pull request of the code that was written or something along those lines to validate that the work delivered meets the engineering standard we have set. Please review the [Developer Review](#) stage gate.

#Research Spikes

The need for a spike is usually identified by the team during our sprint planning sessions and engineering team meetings but it might also arise from other discussions. Once agreed, it is added to our agile task tool along with a description and a list of starter questions to guide areas of research. We categorise a spike as a chore because this story type does not involve a story point estimate. We'll also tag it with the label 'spike' along with project specific labels to keep track of it and distinguish it from other chores.

Please refer to [this article](#) on best practices and usage of Spike research

##Writing and evaluating Research Spikes

Consider the following when writing user stories and/or a research spike:

- Write a clear description of the work that will be conducted and why it is being conducted.
- Definition of "Done"
 - The story is considered "done" when the user can complete the outlined task, but make sure to define what that is.
- Output
 - What are we trying to learn, discover, or achieve with the Spike?
 - The research and findings should be well documented and attached to the story before the SPIKE is moved into Dev Review
- Outline subtasks or tasks
 - Decide which specific steps need to be completed and who is responsible for each of them.
- Time
 - Time is a touchy subject. Many development teams avoid discussions of time altogether, relying instead on their estimation frameworks. Since stories should be completable in one sprint, stories that might take weeks or months to complete should be broken up into smaller stories or should be considered their own epic.

Testing Impact - Does this apply to the spike?

- Test cases and scenarios that may be involved

##Refining and evaluating Research Spikes before sprint commitment Consider the following when reviewing and planning before commitment of this work into a future sprint.

- Does the research spike have a clear understanding of the the expected research and the output of will be reported as a result?

##Reviewing the output of the work conducted within the User story

- Dev Review Stage
 - Research Spikes normally have a Review process in which a team member will review the research documented (usually attached to the spike or linked elsewhere)

#BA Work ##Writing and evaluating BA work items before sprint planning Consider the following when writing these work items before they are moved into the planning ceremony and assigned for sub tasking and estimations

- Write a clear description of the work that will be conducted and why it is being conducted.
- Is there an associated Epic, Feature, User Story, or Bug for this analysis work?
 - Does the associated feature or request have acceptable information for proper requirements gathering?
 - Does the associated feature, epic or request have clearly written expectations of what the result of the requested work will produce?
 - Does this work require engineering resources for requirements gathering, technical analysis, extended consultation and/or review?
 - If yes, please add subtasks with proper time allocated and find a Engineer to assign it to.
- Clear definition around what the resulting output will be
 - Provide information around the expected delivery results of the BA work item.
 - Example: Technical analysis will result in a data dictionary and diagram workflow representing how the data flows for X to Y to Z

##Refining and evaluating before sprint commitment Consider the following when reviewing and planning before commitment of this work into a future sprint.

- Considerations TBD #Child Task

Bugs

Reporting Bugs

Users shall report bugs to BA/QA with the following components:

- Clear reproduction steps starting from homepage

- Observed versus expected observations
- Screenshots of bug or error messages
- Priority
- Severity

The BA will take the following steps:

1. Acknowledge receipt of bug report.
2. Validate bug.
 - If it cannot be reproduced, coordinate with LF team.
3. Enter bug into ADO with all information.
 - Concise title that describes issue.
 - Validated reproduction steps with screenshots.
 - All screenshots should have descriptions.
 - Priority (as reported by LF team)
 - Severity (as reported by LF team).
 - "Tested By" link to relevant Test Case when possible.
4. Respond to tester user with ADO number and link.

Priority Priority should take into account User Stories as well as Bugs. For example, if a Bug is made Priority 1, we should keep in mind that that indicates it will be completed before any Priority 2 User Stories.

Severity

ADO Rank	Level	Description
1	Blocker	No workaround is available; no downstream test scenarios can be conducted until defect is resolved; testing is halted
2	Critical	A core functionality of the system fails or the system does not work at all
3	Major	Defect has impact on basic functionality and the system is unable to produce required results
4	Moderate	Defect causes the system to generate false, inconsistent, or incomplete results
5	Minor	There is impact on the business, but only in a very few cases
6	Cosmetic	Defects that are related to the interface and appearance of the application

Bug States |State| Trigger| Description| |--|--|--| |New| Bug is added to ADO |Bug has been validated by BA/QA and entered into ADO with all needed information| |Active| Developer begins work |Bug fix is being research and/or developed| |Dev Review Developer completes fix Bug fix is being validated by Tech Lead or Dev Manager| |Testing| QA or UAT begins |Testing is in progress; similar to Active, but for QA or UAT team| |Blocked| Cannot make progress| Assigned team member cannot move forward; must leave comment explaining the blocker| |QA Ready| Bug fix is complete and reviewed| Bug is ready for QA testing| |UAT Ready| Bug fix passes QA testing| Bug is ready for UAT testing| |Resolved| N/A| Not in use| |Prod Ready| Bug fix passes UAT testing |Bug fix is ready for deployment to production| |Closed |Bug is deployed to prod| All work is completed and deployed| |Archived| N/A| Not in use| |Removed| Bug is deemed not applicable and/or entered in error| Bug does not need to be reviewed, researched, or worked on; will not show up in sprint or backlog|

Guidelines for Bug creation

Overview: When reporting a bug/defect, it is important to give as much detail to the developer as possible. At the same time, one should try to keep it short and to the point.

What to include in a bug? Guidelines and Example for Bug creation

Overview: When reporting a bug/defect, it is important to give as much detail to the developer as possible. At the same time, one should try to keep it short and to the point. What to include in a bug?

How to open a bug in ADO?

1. From the sprint backlog, click the “New Work Item”
2. From the Drop down that pops up, select “Bug”
3. Give the Description, and click “add to top”
4. Your work item/bug, will show on the top of the work item list. Click the bug and add details using the guideline above.
5. Once you are complete, a bug work item should look something like this:

It is recommended to put the below seven items when reporting a bug:

1. [Feature Name] Title: Your title should serve as a concise summary of what the bug is. Example: If reporting a bug in a report, title can be
2. Environment- If your application has different environment like dev/test/UAT/Prod, enter the information. If applicable, you can enter versions here as well.
3. Steps to reproduce- Document exact steps that should be followed, in order to recreate the bug.
4. Expected Result- What was your expected result when above steps were performed.
5. Actual Result- What is the actual result when the above steps were performed.
6. Visual Proof- screenshots, videos, text etc.
7. Severity/Priority- Sharing the severity offers a degree of impact the bug has on the functionality of the system and helps the dev team prioritize their work. We recommend using one of three categories of severity in your bug report:
 - High/Critical: anything that impacts the normal user flow or blocks app usage
 - Medium: anything that negatively affects the user experience
 - Minor: ALL else (e.g., typos, missing icons, layout issues, etc.)

How to open a bug in ADO?

1. From the sprint backlog, click the “New Work Item”
2. From the Drop down that pops up, select “Bug”
3. Give the Description, and click “add to top”
4. Your work item/bug, will show on the top of the work item list. Click the bug and add details using the guideline above.