MERCURY
PERMITTED CUSTOMISATIONS

Permitted Customisations to Mercury

This document is classified as PUBLIC. It may be disclosed or passed to persons outside the organisation. The latest public version can be found at https://wearemercury.com/permitted-customisations/

This document provides guidelines and rules governing the nature of customisations that are deemed permissible to the Mercury solution.

Contents

1. Purpose, Scope, and Audience

This document outlines the rationale behind why it is necessary to restrict customisations to Mercury. It also provides some guiding principles to encourage “safe” customisation and clearly defines which changes are deemed by Mercury to be permissible.

Customisations are any changes made to Mercury’s system behaviour by parties outside of Mercury’s development team, where those changes are not merely configuration.

The intended audience of this document is anyone outside of Mercury’s internal development team who may be tasked with understanding, specifying, or undertaking changes to the user interface, data model or business logic of the Mercury solution. This audience therefore includes Mercury’s own Consultancy team, customers’ internal IT departments or any other third-party agency or consultant.

2. Associated Documents

There are no associated documents.

3. Glossary of Terms

Term	Description
Customer-owned	Power Platform solutions or solution components that are created for or created by the customers organisation. Importantly this includes any customisations carried out on the customer's behalf by Mercury Consulting or third parties.
Custom Table	Any Table that is not part of the core Dataverse entity model. This includes all Customer-owned Tables and all Mercury-owned Tables with a schema name that starts with any of the following prefixes: • crimadd_ • crimson_ • mercury_ • recruit_
Fork	A fork occurs when a point-in-time clone is made of a solution component, so that it can be further developed independently of the original. Forked components will not inherit future changes from the original component. Once a component has been forked in this way, any future attempt to merge it back with the original is necessarily manual, bespoke and potentially difficult to achieve.
Mercury Core	The main, "core" solution that is deployed into customers' Dataverse environment, that includes non-optional elements of the Mercury product. This appears in a Power Platform environment under (Managed) Solutions as "Mercury Transform"
Mercury-managed	Power Platform solutions or solution components that are deployed by Mercury as part of our Core Solution, Add-On Solutions or official patches. Note that "Mercury-managed" components will therefore include some native Power Platform components such as Tables, Columns and Forms and consequently these components are subject to customisation restrictions.
Permitted customisations	Those customisations to Power Platform solution components that are explicitly listed in this document as being permitted.
Persistent tables	A subset of Mercury-managed **Custom** Tables that can be safely relied upon to always exist. These are defined in APPENDIX 3: Persistent Tables
Unpermitted customisations	Any customisation to Power Platform solution components that are either: · explicitly listed in this document as being **not** permitted for change, OR · not explicitly listed in this document **at all**

4. Overview

Mercury is a sophisticated product, which has Microsoft Power Platform at its core and is integrated with our proprietary set of APIs and web services. As an organisation we have dozens of developers working on changes to Mercury every single week. The underlying platform also benefits from frequent updates thanks to Microsoft's ongoing investment. As a result of this unrelenting rate of change, careful and regimented management is required by our development and quality teams as part of our commitment to delivering an advanced, comprehensive recruitment solution to our customers.

We understand that every recruitment organisation has its unique requirements. As a result, Mercury is designed so that much of our functionality can be adapted through configuration. We also recognise that there may be rare instances where customers feel that specific customisations are important to help optimise their usage of our product. However, to maintain the performance and reliability of our solution, customisations must be performed carefully and judiciously. Inevitably, not all customisations are compatible with our goal of providing and supporting a reliable, high-quality software solution.

This document provides the essential guidelines for customising Mercury. It is designed to support thoughtful, informed customisation so that customers can make effective changes that add value to their business operations, while safeguarding the solution's overall integrity and the quality of the end user experience. These guidelines embody learnings from Mercury's own change management processes and aim to ensure that customisations are undertaken in a manner that preserves system integrity, upholds stability, and minimises unmanageable complexities.

5. Why Guidance is Required

Mercury is built on Microsoft's Power Platform, a robust suite of tools with flexibility at its core. Despite having an inviting "point and click" based interface, safe customisation of Power Platform-based solutions requires a solid understanding of its unique behaviours in certain areas, such as:

- **Solution Layering**: Microsoft Power Platform uses a layering system for solution components, where changes in one layer can impact the other. Unexpected or out of sequence solution layers involving Mercury-managed components have the potential to interfere with routine upgrade cycles and in some cases can cause seemingly innocuous changes to block or undermine the upgrade process.
- **Unmanaged Solution Layers**: Unmanaged solution layers occur when changes are made directly to a production system or where changes are imported contrary to Microsoft's recommended protocol. They are a common cause of system instability and can be especially hard to recognise and isolate.

- **Deprecations and Microsoft's release cadence**: Microsoft issues regular updates to the platform and sometimes deprecates certain features. Customisations need to keep up with this cadence and adapt to ensure that the functionality remains unaffected.
- **Consistency**: There is often more than one way to implement business logic using the Power Platform toolset. Mercury's development teams have spent many years honing their approaches and establishing consistent working practices. If additional business logic fails to follow the same patterns, it introduces additional complexity, particularly when troubleshooting.
- **Code Quality**: Poorly implemented code or logic can impact the normal behaviour or performance of Mercury. Failure to follow best practice and to apply good design thinking can lead to fragile or non-functioning environments.
- **Resource Usage**: Microsoft impose and enforce limitations on API calls which can be particularly challenging for a comprehensive solution such as Mercury. Storage also incurs additional usage-based costs and therefore should be managed carefully. Failure to optimise resource usage can create high impact problems that affect the entire Power Platform tenancy and therefore system stability for Mercury.

Understanding these intricacies and ensuring that customisation efforts respect them is essential to maintaining Mercury's reliability and maximising its value to customers. Non-compliance can lead to unintended consequences such system instability, reduced performance, data loss, and in the worst cases deployments that we are no longer able to support.

These guidelines can only provide a foundation for making informed decisions. At all times, the prevailing guidance from Microsoft should also be considered along with any idiosyncrasies or changes that may exist in the customer's environment. Ill-informed or accidental actions by users, especially those with system administrator level permissions, can cause significant harm to system or data integrity.

6. Support and Maintenance Considerations

6.1. Mercury does not provide support for customisations

While this document describes what Mercury deems to be permissible customisations, customers should be clear that this does not imply that Mercury will provide support for any customisations carried out by 3rd parties. Indeed, even where Mercury has provided consultancy that results in customisations, those changes are only supported where an explicit agreement is in force to this effect.

6.2. "Permitted" is not a guarantee of compatibility

Mercury cannot warrant that customisations carried out in accordance with this document will be guaranteed to work (or be compatible) with Mercury either now or in the future. Adhering to the guidance in this document represents our best source of advice to help 3rd parties minimise ongoing effort in managing customisations alongside Mercury's own updates.

6.3. Unmanaged changes in Production environments are at Customer's risk

Deployment or creation of unmanaged layers in non-development environments is not permitted. Where unmanaged layers are introduced, it is entirely at the Customer's risk.

Mercury upgrades, patches and hotfixes are typically deployed with the "Overwrite Customizations" option selected, which will "overwrite or remove any unmanaged customizations previously performed on components included in this solution".

Where such unmanaged changes interfere with the successful deployment of Mercury's upgrades, patches and/or hotfixes, those changes are liable to unilateral manual removal in the interests of Mercury meeting its obligations under the SaaS Agreement.

6.4. Elevated levels of support will incur charges

Customisations may create elevated levels of support (e.g., during deployment of upgrades, patches and/or hotfixes; or simply through routine use of the software). Such elevated levels of support are liable to incur additional charges. Where fault diagnosis implicates customisations as the root cause, Mercury has no obligation to resolve those issues and will undertake to do so at their sole discretion, at the customer's cost.

6.5. Development Environments

The state and management of the environment that is used to undertake custom development is important and can have long term implications for the ongoing health of the ALM process for both Mercury and the Customer alike.

Development environments should contain the **absolute minimum** number of managed solutions required to undertake the desired development. The goal should not be to attempt to replicate test or production environments. If the development work does not *require* a particular managed solution (3rd party or otherwise) then the solution should not be installed in that development environment. From a Mercury perspective, this means that only Mercury Core should be installed in the development environment save for pre-agreed exceptional circumstances.

7. Core Principles

Due to the flexibility of the underlying platform, for the most part it is not possible to programmatically prevent customers from making customisations to their Mercury deployment. This puts the onus of compliance upon the customer to ensure that where

changes are deemed necessary, they are done so responsibly and within boundaries that are assessed to be “safe” and therefore permitted by Mercury. The core principles should underpin all decisions when considering customisation to Mercury.

7.1. Only make modifications that are explicitly permitted

The main body of this document addresses some specific scenarios and detail; however, the over-arching rule is that you should only consider making customisations of the type that are explicitly permitted in APPENDIX 1: Permitted Customisations.

> ⚠ **Unless a component type is explicitly declared by Mercury to be permitted for customisation, then it should be assumed that customisations are not permitted.**

7.2. Always favour configuration

Customisations should be considered as a last resort. From a system stability perspective, configuration points will always be the better option, even if it means compromising on requirements. Both Mercury and the underlying Power Platform have a range of configuration points that can influence anything from field visibility on forms and setting required fields, to the process for approving Placements or onboarding new candidates. We recommend that you discuss configuration options with your Customer Success Manager before committing to customisation.

7.3. Adopt a formal Change Management process

Naturally, updating the functionality of a live system should not be undertaken until sufficient testing has been carried out. Designing and conducting adequate tests is beyond the scope of this document but it is sufficient to say that tests should consider more than the intended behaviour (“happy path”). Testing should also exercise “unhappy paths”, i.e., the unintended consequences that might occur in the event of adverse scenarios such as missing information, or a user taking a different approach to the specific steps that were tested.

Document your changes in a consistent, searchable format that records the following information at minimum:

- Which component was changed
- Why it was changed (what was the intended benefit)
- When was it changed
- Who made the change

- Any additional steps needed to make it work as expected (e.g., configuration, complimentary components, user training etc.)

Logging changes in this way is the minimum requirement in implementing a change management process and can give valuable insight to help you troubleshoot problems by helping you identify which components have been modified.

7.4. Follow Best Practice for Solution Management

NEVER CREATE UNMANAGED LAYERS IN NON-DEVELOPMENT ENVIRONMENTS

Changes should **always** originate in a development (non-production) environment. Customisations made directly to a production environment are not only risky, but they also inherently introduce unmanaged layers which can be very disruptive to ongoing system integrity.

Use a custom Publisher record (i.e., do not reuse any of the Mercury publisher records). This is critical in ensuring a clear distinction between custom components and core Mercury-managed components. The Publisher record should include a "prefix" value that makes it clear where the change originated from. Typically, this will involve the name of the organisation that is subscribed to Mercury.

Solution packages should only include components that are custom or have been customised. Therefore when packaging changes to existing Tables for example, customizers should only include the new or changed components, i.e., do not select "include all entity metadata".

Changes should always be imported to non-development environments as managed solutions. This is consistent with Microsoft's Best Practice guidance and provides benefits that include predictable component layering and rollback capabilities.

Per Microsoft's advice, we strongly recommend that Solution Patches are not used.

7.5. Do not change or disable Mercury-managed components

Unless explicitly stated in this document, Mercury-managed components should not be changed or disabled except as the result of an upgrade or patch by us.

Therefore, the following Mercury-managed components are among those which **must not be changed or disabled**:

- Web Resources (importantly, this includes all JavaScript files)
- Entity model metadata:

    - Table-level metadata (e.g., auditing, document management etc)
    - Column-level metadata (e.g., auditing, field lengths, option-set values)
    - System Views
- Processes & business logic
    - Classic workflows (except where these have been implemented locally from a Workflow Template)
    - Actions
    - Business Rules
    - Cloud Flows
    - Commands
    - Component Libraries
    - Custom APIs
    - Plug-Ins and related registrations
    - Workflow Assemblies
- Model Driven & Canvas Apps (except where otherwise stated in this document)
- Custom Pages
- Ribbon or command-bar buttons, Modern Commands
- Form specifics
    - Existing Mercury-managed form components within Mercury forms
    - Existing form event registrations within Mercury Forms

7.6. Avoid solution dependencies on Mercury-managed components

Except for those entities that are declared as Persistent Entities (see APPENDIX 3: Persistent Tables), Mercury-managed components cannot be guaranteed to always persist between releases and therefore should not be relied upon to always exist. Although it rarely occurs, Mercury retains the right to remove, re-purpose or significantly modify any component within our solutions and as such, your customisations should not create dependencies on them. If such dependencies are created, future attempts to upgrade your environment risk failure. This could result in unilateral removal of your changes, service disruption for your organisations and/or financial penalties for remedy.

Some component types are more susceptible to risk than others. For example, while Tables are unlikely to be removed from one release to the next, it is not impossible (Mercury previously used a custom Candidate table but since migrated to use the native Contact table). Other component types are much more likely to change or be removed without notice, particularly those that handle business logic such as JavaScript files, Power Automate Cloud Flows and Custom Workflow Assemblies. Similarly, some user interface components are susceptible to change such as Quick View Forms, Quick Create Forms, System Views, PCF Controls, and web resource such as images and icons.

7.7. Package Entity Model changes separately

Given the risks to future Mercury upgrades, we strongly recommend that when customisations to Mercury are packaged, all changes to the Entity Model (New columns, new relationships, etc.) are added into their own solution and are deployed separately to any business logic or user interface changes.

By separating entity model from everything else, it is easier to minimise the risks of data loss in the event of customisations needing to be rolled back or uninstalled.

7.8. Consider future updates and support

Monitor Microsoft Updates: Stay aware of Microsoft's release cadence, deprecations, and updates to ensure customisations continue to function correctly.

Consult Support (& release) Documentation: Refer to the support resources provided by Microsoft and Mercury alike. Consider engaging with Mercury's support team if questions arise, to ensure that customisations align with our solution architecture.

> Be aware that by opting to customize a Mercury deployment, you create an obligation to maintain those changes to ensure that they remain compatible in future. This typically involves ongoing monitoring of announcements from both Mercury and Microsoft, and reacting accordingly to address changes in behaviour and deprecations that are outside of your control.

7.9. Do not reference "xDNU" components

Components prefixed with the string "xDNU" or "x DNU" represent items that are deprecated and will be removed in a future release of Mercury. Do not reference them.

8. Rules for creating customer-owned components

8.1. Data Model

It is generally safe to create and maintain brand new data components such as Tables, Columns and Views, however per section 7.6 above we do not permit:

- adding new relationships (lookups or many-many relationships) that involve Mercury-managed tables (except Persistent Tables).
- including Mercury-managed columns in calculated columns (except where the calculations only involve Persistent Tables).
- including Mercury-managed tables or columns in rollup columns (except where the rollups only involve Persistent Tables).
- adding new columns on Mercury-managed tables that have a requirement level of "Required"

Changes made to Mercury-managed Table-level metadata (typically enabling Auditing or Document Management) are prone to be reset after each Mercury patch or upgrade and are therefore not permitted.

> Adding “Required” columns on Mercury-managed tables can mean that newly-deployed Cloud Flows that are part of the Mercury solution are not able to be turned on after deployment.

8.2. User Interface

It is generally safe to create and maintain new customer-owned UI components such as Tabs, Sections, Form Controls and Columns. Notwithstanding the dependency principle outlined in section 7.6 above, it is also permissible to create new customer-owned forms against Mercury-managed entities however customizers should pay particular attention to section 10 Considerations for Forms. It should also be noted that by adopting a custom form, customers potentially lose the benefit of Mercury’s ongoing investment while also introducing a maintenance burden for themselves.

New customer-owned views and dashboards are also permitted, despite the risk they introduce as outlined in section 7.6 above. Customers should be aware however that in the unlikely event that such customer-owned dashboards or views prevent upgrade of a Mercury installation, those components that are preventing the upgrade are liable to be unilaterally deleted.

8.3. Business Logic

Mercury considers all new business logic to be permitted, except where that business logic exhibits one or more of the following features:

- Disabling or over-riding of any Mercury business logic (except where this is the result of a Mercury configuration value that has been explicitly documented). This restriction applies to all business logic, including:
  - Ribbon Buttons & Commands
  - Form event registrations
  - JavaScript libraries
  - Plug-Ins and their associated assemblies, steps, and images
  - Processes
    - Actions
    - Business Rules
    - Classic Workflows
    - Cloud Flows
- Introduction of a Solution Dependency on a Mercury-managed:
  - Table

    - Column
    - Action
    - Custom API
    - Custom Workflow Assembly
    - Web Resource (esp. JavaScript files)
- Overwriting non-null data in Mercury-managed tables, especially in columns that represent statuses or status reasons.
- Registering new plugin-in steps using Mercury-managed plugin assemblies.
- Implementing synchronous logic on a Mercury-managed custom table.
- Implementing synchronous logic on one of the following native tables:
    - Client (account)
    - Contact (contact)
    - Note (annotation)
    - User (systemuser)
    - Team (team)
    - Address (customeraddress)
    - Any Activity table (activitypointer) including but not limited to:
        - Appointment (appointment)
        - Email (email)
        - Task (task)
- Switching mandatory Mercury-managed form controls to be non-mandatory.

Customizers that are considering changes or additions to Business Logic should also read section 11 Considerations for Business Logic.

9. Considerations for Data Model

9.1. Tables

Metadata at the Table level for Mercury-managed Tables is not permitted for modification. This includes display labels and toggling of settings that relate to auditing and document management. Such changes are likely to be automatically rolled back after each Mercury upgrade or patch.

9.2. Relationships

New relationships that involve Persistent Tables or Customer-owned Tables are permitted however relationships that involve any other Mercury-managed Tables must not be created.

9.3. Calculated Fields

New calculated fields are permitted so long as they only reference columns on Mercury-managed Persistent Tables or Customer-owned Tables.

9.4. Rollup Fields

New rollup fields are permitted so long as they only reference columns on Mercury-managed Persistent Tables or Customer-owned Tables.

9.5. Columns (Attributes)

New columns are permitted on Mercury-managed Persistent Tables or Customer-owned Tables however they **must not** be set with a Requirement Level of “Required”. If a custom column is set to “Required” at the Table (Entity) level, then any Cloud Flows that Mercury may deploy with our solution will not be able to be turned on without further customisations.

> If a new mandatory column is needed on a Mercury-managed table, the we recommend using Client-side code or form-scoped Business Rules to set the field requirement level at runtime. This provides the business benefits of mandatory field, without the deployment restrictions caused by setting this at the Table level.

10. Considerations for Forms

10.1. Mercury-managed Controls, Sections and Tabs

- Mercury-managed form controls should never be removed from forms. This creates an untested configuration that can adversely affect system performance and data integrity.
- Changes to the properties of Mercury-managed form controls (including visibility, labels and formatting) are not permitted.
- Mercury-managed form sections should not be moved, removed, hidden, or renamed. This is an untested configuration and can result in unexpected behaviour in the user interface. Sections that have been compromised are likely to be “reset” to their original location after each Mercury update, in any case.
- Mercury-managed form tabs should not be moved, removed, hidden, or renamed. This is an untested configuration and can result in unexpected behaviour in the user interface. Tabs that have been compromised are likely to be “reset” to their original location and configuration after each Mercury update in any case.

10.2. Customer-owned Controls, Sections and Tabs

- The recommended approach for showing Customer-owned form controls is to add them to a customer-owned section within a Customer-owned tab. Sections and tabs should be named with a customer-specific prefix to minimise the chances of “collisions” with current or future Mercury-managed components.
- It is not recommended to add Customer-owned form controls to Mercury-managed form sections as they are not guaranteed to remain in the same location after each Mercury update. Customer-owned form controls are more consistent if added to new customer-created form sections.
- It is not recommended to add Customer-owned form sections to Mercury-managed form tabs, since their position within the tab layout cannot be guaranteed to persist each Mercury update.

10.3. Merging & Merge Conflicts

When deployed into a production environment as part of a managed solution, form changes that follow our recommendations can typically be expected to "merge" safely with Mercury behaviour. However, situations can arise whereby customisations cannot be merged and instead cause a conflict. Should this occur, the deployment process will result in a new "Conflicts Tab" being created on the form, containing any problematic form components. These situations typically occur when two different solutions contain conflicting information (e.g., a newly introduced customer-owned form tab that has the same name as a Mercury-managed tab – see section 10.2 for guidance on avoiding this).

The preferred resolution is to revert to the Customer's development environment and update the Customer-owned solution to remove the conflict (e.g., rename the form tab, section, or control) and then redeploy the new managed solution. Note that due to risks of data loss we advise against simply uninstalling the Customer-owned managed solution.

There is little inherent risk in the Conflicts tab remaining and not being resolved, but it is to the detriment of the user experience for end users.

10.4. Cloning Forms

Cloning of Mercury-managed forms is not permitted. This approach would create a fork from the Mercury solution (i.e., a snapshot copy) which means many of the form-level investments that we may carry out in future will not be reflected on the cloned form. Further, references to our script libraries will also remain in the cloned forms. Future changes to these JavaScript libraries (including decoupling them from our forms) by Mercury could lead to upgrade failures to the solution, or runtime errors whereby requisite components do not exist on the cloned form.

10.5. New forms

Creating brand new customer-owned forms for Mercury-managed tables is not recommended. Using new forms carries a high risk to system integrity as it can readily result in untested scenarios and errors in our Mercury business logic, due to missing, conflicting or ambiguous data.

10.6. Form Event Registrations

Changing or removing Mercury-managed form event registrations is not permitted as it undermines system integrity.

While adding new form event registrations is permitted, it should be noted that it is risky. Issues such as poor syntax can readily cause existing Mercury client-side code to fail. It is therefore important that client-side scripts are written accurately and defensively to reduce the risks of knock-on failure.

11. Considerations for Business Logic

Notwithstanding the restrictions outlined in section 8.3 Business Logic, those customisations that are permitted should always consider the following points.

11.1. API Limits

Be mindful of the API limits that are published by Microsoft and may change from time to time. Since breaches of Microsoft's API limits typically result in "throttling" (restricting) of service availability, inefficient custom logic can adversely affect the operation of Mercury by effectively "locking out" the Mercury business logic until service is restored.

11.2. Avoid Indiscriminate Triggers and Unnecessary Updates

When registering new business logic, be sensitive to which columns will trigger your logic. If your logic fails to discriminate (or does so inefficiently), then it can cause unnecessary processing which, in the worst cases, can cascade and trigger other logic to run unnecessarily.

Similarly, where custom logic contains updates to data, it is important to avoid "updating" a column where the data in that column has not actually changed. This can cause Mercury-managed business logic to be triggered which, in the worst cases, can result in compromised system stability or data integrity. These issues can only arise when custom business logic is introduced, as the native user interface otherwise prevents this from happening.

11.3. Power Automate / Cloud Flows

Changes announced by Microsoft in July 2023 mean that automated Power Automate Flows are susceptible to being unilaterally disabled unless they are correctly "associated" with a known Canvas- or Model-Driven-App. Consequently, Mercury recommends that you familiarise yourself with the process of associating Apps and Flows with each other.

12. Security Roles and Column Security Profiles

Neither Mercury-managed Security Roles nor Mercury-managed Column Security Profiles are permitted for customisation. Since security privileges in Dynamics 365 are summative (i.e., a user's net privileges are the sum of all privileges in all roles that they are party to), customers may extend privileges by creating new roles and assigning them alongside the Mercury-managed security roles.

12.1. Column Security Profiles

For a column to be subject to Column Security Profiles, it is necessary to first update the metadata on that column to state that it supports Column Security. Making such changes to Mercury-managed columns is not permitted, however creating a new Column Security Profile that references Mercury-managed columns that are already enabled for Column Security *is* permitted.

12.2. Security Roles

Mercury recommend that customers opt to create discrete roles for additional privileges (i.e., create new Security Roles which can be used to supplement Mercury's out of the box roles) rather than cloning Mercury roles and amending them. Mercury's out of the box Security Roles are used to manage access to our Model Driven- and Canvas Apps and are also referenced in some ribbon level and form level business logic to influence user experience. Cloning Mercury's Security Roles (and allocating those roles to your users instead) would result in an unsupported user experience. Further, ongoing investments in Mercury often result in privileges being appended or removed from roles and these updates would not be propagated to any (cloned) custom roles.

13. Power Apps, Site Maps and Ribbons

13.1. Canvas Apps & Custom Pages

Mercury-managed Canvas Apps and their counterpart, Custom Pages, are explicitly not permitted for modification.

13.2. Model Driven Apps

The Mercury product comes with a minimum of two Model Driven Apps ("Mercury", and "Mercury Admin"). These Apps can only display data from those Tables defined within the

App by the Mercury product development team. It is not possible for a customer to expose data from additional Tables within this App unless the Model Driven App is customised. **However, customisations to the Mercury and Mercury Admin model driven apps are strongly discouraged.**

Should it be deemed necessary to include an additional Table within a Mercury App then customers may be tempted to fork (i.e., clone and then customise) the original App. Mercury recommend against this approach as it introduces several issues:

- The customer-owned fork will never receive future updates to the Model Driven App
- Additional administration of the customer-owned copy App is required to ensure that end users have the appropriate Security Roles to access the new App
- There may be licence issues to consider if users are granted access to another App

One suggested compromise in this scenario is to deploy managed customisations to the Mercury Model Driven App but accepting that this process will need to be repeated after each of Mercury's upgrades.

13.3. Sitemap

Existing Mercury-owned Sitemap entries are not permitted for customisation, however it is permitted to create new Sitemap entries for customer-owned entries as they will merge when deployed.

13.4. Command Bar / Ribbons / Buttons

Dataverse currently provides two options (classic and modern) for adding ribbon buttons to the UI. Classic commands are customised using direct XML manipulation or 3rd party tools such as Ribbon Workbench. **Mercury does not support ribbon changes using classic commanding** as they exhibit poor solution layering behaviour that can mask or override Mercury's ribbon buttons**.**

> Classic command bar behaviour depends on solution layering at the button group level, not at the individual button level. The highest active customisation layer that includes a given ribbon button group determines the complete definition of that group. It does not merge buttons across layers.
>
> For example, if layer 1 defines a group containing ButtonA, layer 2 defines the same group containing ButtonA and ButtonB, and layer 3 defines the group containing only ButtonC, then the effective outcome will be ButtonC only. The definition in layer 3 replaces the group entirely.

As a result, any modification to ribbon button groups used within Mercury carries a significant deployment risk, as introducing a higher layer that redefines the group can unintentionally remove previously configured buttons.

Modern Commands can be customised from within the Power Platform UI. Modern commanding can consume functions in JavaScript Web Resources, similar to the classic approach via Ribbon Workbench or using Power Fx. **Mercury supports modern commands, but only where they consume customer-owned JavaScript web resources.**

Power Fx based actions are not permitted as there can only be a single command component library for each (model-driven) App, and those component libraries do not support merge behaviour, therefore changes to this library will mask or override core Mercury functionality

Known limitations of Modern Commanding are documented by Microsoft. At the time of writing, the most notable of these limitations is an inability to toggle visibility of a button.

Should it be necessary to restrict access to the business logic behind a Modern Command button, Mercury recommend that checks are made within the JavaScript logic to determine the necessary permissions or access conditions.

13.4.1 Dynamic Ribbon Buttons

Mercury version 2026.2 (v37) introduced the concept of fully dynamic ribbon buttons, which will gradually roll out across our suite of solutions. This approach introduces a new “Mercury” flyout button to forms and sub-grids, under which all Mercury buttons are dynamically loaded using data in two new tables (`recruit_customribbonbutton` and `recruit_customribbonsection`). This uses classic ribbon buttons and introduces a new ribbon button group on each ribbon.

Mercury do permit new rows to be added in the both new tables, which may support customer use cases for custom buttons, however we do not currently provide any support for such customisations.

It is strictly prohibited to edit this new group or the button within it. Due to its critical role in the product, where unpermitted customisations to this component are detected, Mercury will unilaterally remove them to protect system stability. Developers

considering making changes to these components are reminded not to do so as the group name is prefixed `DO_NOT_EDIT`

APPENDIX 1: Permitted Customisations involving Mercury-managed components

Customisations that involve (i.e. reference either directly or indirectly any Mercury-managed component including but not limited to references in code, processes, entity relationships, user interface metadata, and solutions) Mercury-managed Components but are not explicitly included in this Appendix are deemed by Mercury to be unpermitted. Customers should be aware that undertaking such unpermitted customisations:

- carries significant risk to system integrity, data integrity and consistent user experience.
- introduces high risk of breakage or loss whenever Mercury conducts routine or emergency upgrades and patches.
- can result in remediation fees, if those customisations are found to cause or contribute to excessive technical support effort.

Legend

✅ Permitted	Change is permitted and can be expected to persist despite maintenance by Mercury
⚠️Permitted with warning	Change is permitted but creates a dependency on a Mercury-managed component, therefore in the unlikely event that this change prevents Mercury from upgrading, patching or otherwise maintaining your environment this change is liable to be unilaterally removed. If the advice in section 7.7 (Package Entity Model changes separately) has been followed then no data loss will occur as a result, however your change may need to be reapplied by you after our maintenance is completed.
🚩 Permitted with exceptions	See the Exceptions and Clarifications columns to understand the limits of what is permitted
	If no wording or colour is shown, this change is not permitted

Component Type	Create / modify non-Mercury components	Modify Existing	Remove/Disable	Exceptions	Clarifications
Entity Model					

Table	✅ Permitted	⛔ Not permitted	⛔ Not permitted		No changes permitted. Includes changes to entity-level metadata such as Auditing or Document management See section 9.1 Tables
Column	🚩Permitted with exceptions	⛔ Not permitted	⛔ Not permitted	Lookup columns are not permitted unless they only involve Persistent Tables or Customer-owned Tables Calculated and Rollup Columns have specific restrictions, see section 9.3 and section 9.4 Custom columns on Mercury-managed entities must **not** have “requirement level” set to “Required”. See section 9.5	Permits creation of certain new columns on Mercury-managed entities
Relationship	🚩Permitted with exceptions	⛔ Not permitted	⛔ Not permitted	New relationships are not permitted unless they only involve Persistent Tables or Customer-owned Tables	Permits creation of certain new relationships on Mercury-managed entities
Key	⚠️Permitted with warning	⛔ Not permitted	⛔ Not permitted		See legend

User Interface					
Form		🚩Permitted with exceptions		See section 10 Considerations for Forms	
Personal Views	⚠️Permitted with warning	⚠️Permitted with warning	⚠️Permitted with warning		See legend
System Views	⚠️Permitted with warning	⛔ Not permitted	⛔ Not permitted		See legend
Ribbon components	✅ Permitted	⛔ Not permitted	⛔ Not permitted		
Model Driven Apps					
Model Driven Apps	✅ Permitted		⛔ Not permitted	See 7.6 Avoid solution dependencies on Mercury-managed components	No changes permitted to Mercury-owned Apps. See below for permitted changes to components within Mercury-owned Apps
Forms	✅ Permitted				Can include new Forms
Sitemap	✅ Permitted				Can include new sitemap entries
System Views	✅ Permitted				Can include new System Views
Tables	✅ Permitted				Can include additional Tables
Modern Commands	🚩Permitted with exceptions			Use of customer-owned JavaScript Web Resources only. Do not use Power Fx.	See 13.4 Command Bar / Ribbons / Buttons
Canvas					
Canvas Apps	✅ Permitted	⛔ Not permitted	⛔ Not permitted		No changes permitted to

					Mercury-owned components
Custom Pages	✅ Permitted	⛔ Not permitted	⛔ Not permitted		No changes permitted to Mercury-owned components
Other					
Column Security Profiles	⚠️Permitted with warning	⛔ Not permitted	⛔ Not permitted	See section 12.1 Column Security Profiles	Can create new Column Security Profiles that involve Mercury-managed components. See legend.
Security Roles	⚠️Permitted with warning	⛔ Not permitted	⛔ Not permitted	See 7.6 Avoid solution dependencies on Mercury-managed components	See legend See 12.2 Security Roles
Client-Side Business Logic					
Business Rules	🚩⚠️Permitted with warnings and exceptions	⛔ Not permitted	⛔ Not permitted	See section 8.3 Business Logic	
Form Event Registrations	✅ Permitted	⛔ Not permitted	⛔ Not permitted		
JavaScript	🚩⚠️Permitted with warnings and exceptions	⛔ Not permitted	⛔ Not permitted	See section 8.3 Business Logic	See section 11 Considerations for Business Logic
Server-Side Business Logic					

Actions & Classic Workflows	🚩⚠️Permitted with warnings and exceptions	⛔ Not permitted	⛔ Not permitted	See section 8.3 Business Logic	See section 11 Considerations for Business Logic
Cloud Flows	🚩Permitted with exceptions	⛔ Not permitted	🚩Permitted with exceptions	See section 8.3 Business Logic	See section 11 Considerations for Business Logic
Custom API	🚩⚠️Permitted with warnings and exceptions	⛔ Not permitted	⛔ Not permitted	See section 8.3 Business Logic	See section 11 Considerations for Business Logic
Plug-in Assemblies	🚩⚠️Permitted with warnings and exceptions	⛔ Not permitted	⛔ Not permitted	See section 8.3 Business Logic	See section 11 Considerations for Business Logic
Plug-in Steps	🚩⚠️Permitted with warnings and exceptions	⛔ Not permitted	⛔ Not permitted	See section 8.3 Business Logic Not permitted to create new Plug-In Steps that involve Mercury-managed Plug-In Assemblies	See section 11 Considerations for Business Logic
Plug-in Images	🚩⚠️Permitted with warnings and exceptions	⛔ Not permitted	⛔ Not permitted	See section 8.3 Business Logic Not permitted to create new Plug-In Images that involve Mercury-managed Plug-In Steps	
Workflow Assemblies	🚩⚠️Permitted with warnings and exceptions	⛔ Not permitted	⛔ Not permitted	See section 8.3 Business Logic	See section 11 Considerations for Business Logic Do not reference Mercury workflow

					activities as they are not guaranteed to persist between releases.
Workflow Templates	🚩⚠️Permitted with warnings and exceptions	🚩Permitted with exceptions	🚩Permitted with exceptions	See section 8.3 Business Logic	

APPENDIX 2: Mercury-managed Tables

Mercury-managed Tables are those Tables that are deployed by Mercury as part of our Core Solution, Add-On Solutions, or patches. While broadly static, this list typically changes with each release of Mercury with additions being generally more likely than removals.

Mercury-managed tables includes **all tables with the following schema prefixes**, plus several core/native Tables:

- crimadd_
- crimson_
- mercury_
- recruit_

The prevailing list of Mercury-managed tables can be confirmed independently by following this process:

1. Go to the Power Apps maker portal at Power Apps
2. Login using an account with customisation privileges.
3. Go to Solutions in the left-hand navigation.
4. In the Solutions list, filter the Publisher column such that it includes all rows that contain the term “Mercury”
5. For each solution, check Objects > Tables
    a. Any Table listed here is deemed “Mercury-managed”

As of Mercury v27 this is the complete list of Mercury-managed tables in the core product only:

Display Name	Schema Name	Type
Accounting Legal Entity	mercury_accountinglegalentity	Mercury
Action Card	ActionCard	Native
Activity	ActivityPointer	Native

Activity Outcome	recruit_activityoutcome	Mercury
Activity Purpose	mercury_activitypurpose	Mercury
Address	CustomerAddress	Native
Advert	mercury_advert	Mercury
Agency	mercury_agency	Mercury
Agreement	mercury_agreement	Mercury
Agreement Benefit	mercury_agreementbenefit	Mercury
Agreement Cost Centre	mercury_agreementcostcentre	Mercury
Agreement Expense Rule	mercury_agreementexpenserule	Mercury
Agreement Queue Item	mercury_agreementqueueitem	Mercury
Agreement Rate Type	mercury_agreementratetype	Mercury
Applicant	mercury_applicant	Mercury
Applicant Substatus	recruit_applicantsubstatus	Mercury
Appointment	Appointment	Native
Approval	recruit_approvalactivity	Mercury
Approval Step	recruit_approvalstep	Mercury
Audit event	recruit_auditevent	Mercury
Audit event detail	recruit_auditeventdetail	Mercury
Bank Account	mercury_bankaccount	Mercury
Benefit	mercury_benefit	Mercury
Bulk SMS Message	mercury_bulksmsmessage	Mercury
Business Unit	BusinessUnit	Native
Campaign Activity	CampaignActivity	Native
Campaign Response	CampaignResponse	Native
Candidate Company	crimson_candidatecompany	Mercury
Candidate Role	mercury_candidaterole	Mercury
Candidate Sentiment	mercury_candidatesentiment	Mercury
Client	Account	Native
Client Category	mercury_clientcategory	Mercury
Client Cost Centre	mercury_clientcostcentre	Mercury
Client Project	mercury_clientproject	Mercury
Compliance Criterion	mercury_compliancecriterion	Mercury

Compliance Evidence	mercury_complianceevidence	Mercury
Compliance Group	mercury_compliancegroup	Mercury
Compliance Status	mercury_compliancestatus	Mercury
Compliance Status List Item	mercury_compliancestatuslistitem	Mercury
Configuration	crimson_configuration	Mercury
Configuration Category	mercury_configuration_category	Mercury
Connection	Connection	Native
Contact	Contact	Native
Contact Role	mercury_contactrole	Mercury
Contract	Contract	Native
Conversation	msdyn_ocliveworkitem	Native
Copilot Transcript	msdyn_copilottranscript	Native
Country	mercury_country	Mercury
Country Translations	mercury_countrytranslations	Mercury
Currency	TransactionCurrency	Native
Customer Voice alert	msfp_alert	Native
Customer Voice survey invite	msfp_surveyinvite	Native
Customer Voice survey response	msfp_surveyresponse	Native
CV Document	recruit_cvdocument	Mercury
CV Education	recruit_cveducation	Mercury
CV Parsing Engine	recruit_cvparsingengine	Mercury
CV Skill Level	recruit_cvskilllevel	Mercury
CV Work History	recruit_cvworkhistory	Mercury
Data Consent	mercury_dataconsent	Mercury
Data Consent Change	mercury_dataconsentchange	Mercury
Data Consent Purpose	mercury_dataconsentpurpose	Mercury
Discipline	mercury_discipline	Mercury
Dynamic Email	mercury_dynamicemail	Mercury
Dynamic Email SharePoint Link	mercury_dynamicemailsharepointlink	Mercury
Dynamic Email Template	mercury_dynamicemailtemplate	Mercury
Education	crimson_education	Mercury

Email	Email	Native
Email Counter	mercury_emailcounter	Mercury
Email Limit Definition	mercury_emaillimitdefinition	Mercury
Email Limit Tracker	mercury_emaillimittracker	Mercury
Email restriction	mercury_emailrestriction	Mercury
Email Salutation Mapping	mercury_emailsalutationmapping	Mercury
Engagement	mercury_engagement	Mercury
Error	crimson_error	Mercury
Ethnicity	recruit_ethnicity	Mercury
Expense	recruit_expense	Mercury
Expense Activity	recruit_expenseactivity	Mercury
Expense line	recruit_expenseitem	Mercury
Expense Rejection Reason	recruit_expenserejectionreason	Mercury
Expense Template	recruit_expensetemplate	Mercury
Expense Type	mercury_expensetype	Mercury
Expense Validation Code	recruit_expensevalidationcode	Mercury
Fax	Fax	Native
Fee	mercury_fee	Mercury
Fee calculation	recruit_feecalculation	Mercury
Finance Sync Status	mercury_financesyncstatus	Mercury
Financial Transaction	mercury_transaction	Mercury
Financial Transaction Type	mercury_transactiontype	Mercury
FX Spot Rate	mercury_fxspotrate	Mercury
Hot List	mercury_hotlist	Mercury
Incremental Numbering	crimadd_incrementalnumbering	Mercury
Info	mercury_info	Mercury
Info Shortlist	recruit_infoshortlist	Mercury
Internal Work Item	mercury_internalworkitem	Mercury
Invoice Instruction	mercury_invoiceinstruction	Mercury
Language	recruit_language	Mercury
Letter	Letter	Native
Local Search Update	mercury_localsearchupdate	Mercury

Nationality	recruit_nationality	Mercury
Note	Annotation	Native
Off Limits Reason	mercury_donotheadhuntreason	Mercury
Payment Term	mercury_paymentterm	Mercury
Phone Call	PhoneCall	Native
Placement	crimson_placement	Mercury
Placement Approval	mercury_placementapprovalactivity	Mercury
Placement Approval Step	mercury_placementapprovalstep	Mercury
Placement Benefit	mercury_placementbenefit	Mercury
Placement Cost Centre	mercury_placementcostcentre	Mercury
Placement Custom Field	mercury_placementcustomfield	Mercury
Placement Custom Field Value	mercury_placementcustomfieldvalue	Mercury
Placement Expense Rule	mercury_placementexpenserule	Mercury
Placement Rate	crimson_rate	Mercury
Portal Role	recruit_PortalRole	Mercury
Portal Theme	recruit_portaltheme	Mercury
Position	Position	Native
Process Session	ProcessSession	Native
Project	recruit_project	Mercury
Purchase Order	crimson_purchaseorder	Mercury
Queue Item	QueueItem	Native
Rate Frequency	crimson_ratefrequency	Mercury
Rate Type	crimson_ratetype	Mercury
Recurring Appointment	RecurringAppointmentMaster	Native
Reference	mercury_reference	Mercury
Religion	recruit_religion	Mercury
Report Template	crimson_reporttemplate	Mercury
Report Template Query	crimson_reporttemplatequery	Mercury
Reporting Business Unit	mercury_reportingbusinessunit	Mercury
Reporting Region	mercury_reportingregion	Mercury

ReportTrigger	crimson_reporttrigger	Mercury
Residency Status	mercury_residencystatus	Mercury
Rules of Engagement	recruit_rulesofengagement	Mercury
Saved Query	mercury_searchrequest	Mercury
Search Personalisation	mercury_searchpersonalisation	Mercury
Search Response	mercury_searchresponse	Mercury
Search Result Marker	mercury_searchresultmarker	Mercury
Service Activity	ServiceAppointment	Native
Shortlist	crimson_vacancycandidate	Mercury
Shortlist Feedback	mercury_shortlistfeedback	Mercury
Shortlist Method	mercury_shortlistmethod	Mercury
Shortlist Substatus	mercury_shortlistsubstatus	Mercury
Shortlist Update	mercury_shortlistupdate	Mercury
Situation	mercury_situation	Mercury
Skill	crimson_skill	Mercury
Skill Level	crimson_skilllevel	Mercury
SLA KPI Instance	SLAKPIInstance	Native
SMS Message	mercury_smsmessage	Mercury
SMS Originator	mercury_smsoriginator	Mercury
Social Activity	SocialActivity	Native
Social Network	mercury_socialnetwork	Mercury
Social Profile	mercury_socialprofile	Mercury
Source	crimson_candidatesource	Mercury
Spec Send	mercury_speculativesend	Mercury
Standard Classification	recruit_standardclassification	Mercury
State or Province	mercury_state	Mercury
Subsector	mercury_subsector	Mercury
Sync Error	SyncError	Native
Tag	mercury_tag	Mercury
Task	Task	Native
Tax Code	mercury_taxcode	Mercury
Team	Team	Native

Teams chat	chat	Native
Timesheet	mercury_timesheet	Mercury
Timesheet Activity	mercury_timesheetactivity	Mercury
Timesheet Frequency	recruit_timesheetfrequency	Mercury
Timesheet Line	mercury_timesheetline	Mercury
Timesheet Rejection Reason	mercury_timesheetrejectionreason	Mercury
Timesheet Template	mercury_timesheettemplate2	Mercury
Timesheet Validation Code	mercury_timesheetvalidationcode	Mercury
Update	mercury_update	Mercury
User	SystemUser	Native
Vacancy	crimson_vacancy	Mercury
Vacancy Benefit	mercury_vacancybenefit	Mercury
Vacancy Lost Reason	mercury_vacancylostreason	Mercury
Vacancy Type	mercury_vacancytype	Mercury
Video Call Type	recruit_videocalltype2	Mercury
Work History	crimson_workhistory	Mercury
Work Pattern	mercury_workpattern	Mercury
Workflow Schedule	mercury_workflowschedule	Mercury
X DNU Video Call Type	recruit_videocalltype	Mercury

APPENDIX 3: Persistent Tables

The following Mercury-managed tables are deemed by Mercury to be persistent and therefore highly unlikely to ever be deprecated from our solution. Some types of customisations are only permitted against Persistent Tables.

Display Name	Schema Name	Comments
Client	account	Native to Dataverse
Contact	Contact	Native to Dataverse
Placement	crimson_placement	
Vacancy	crimson_vacancy	

Shortlist	crimson_vacancycandidate	

Version History

Version History

Version	Date	Comment
Current Version **(v. 21)**	**Mar 02, 2026 11:10**	: Added additional detail for ribbon button groups and dynamic ribbon buttons.
v. 20	Mar 02, 2026 11:09	
v. 19	Jan 17, 2025 18:22	Declare restrictions on introducing dependencies on Mercury-owned Web Resources
v. 18	Nov 05, 2024 13:38	Added explicit "not permitted" options to the Appendix for some previously-blank boxes. Switched edits to Cloud Flows to "Not Permitted" (was Permitted with Warnings)
v. 17	Oct 03, 2024 16:42	Minor formatting changes only
v. 16	Jul 09, 2024 09:33	Remove duplicated wording in 11.1
v. 15	Jun 28, 2024 15:22	Added 9.5 which restricts setting columns on Mercury-managed tables to "Required"
v. 14	May 24, 2024 12:13	Added link to public version of the document
v. 13	Feb 23, 2024 16:39	Minor adjustments to wording in section "6.5 Development Environments"
v. 12	Feb 08, 2024 18:56	Definition for "Mercury Core" added to glossary. Point 6.5 added to address some details of development environments.
v. 11	Jan 04, 2024 18:07	Clarified wording in Appendix 1 table, including column header text and references to Mercury Apps versus components within Mercury Apps. Moved Version History to the end of the document for improved readability upon export.
v. 10	Dec 14, 2023 15:51	Added "PUBLIC" document classification
v. 9	Dec 14, 2023 15:45	Fix minor formatting issues after original import from Word
v. 8	Dec 14, 2023 09:58	Minor correct to section header
v. 7	Dec 06, 2023 18:52	Clarifying wording around solution management and inclusion of changed solution components only. Clarification that hiding of Mercury form components is not permitted.

		Clarification of our stance on customising the Mercury app (to address some potential contradictions). Refinement of the term "Persistent Tables".
v. 6	Nov 20, 2023 13:38	Minor correction to wording in 6.3
v. 5	Nov 17, 2023 20:07	Added 7.9 - do not reference xDNU components
v. 4	Nov 17, 2023 20:02	Minor updates to remove unwanted content
v. 3	Nov 17, 2023 19:54	Move version history to expander section
v. 2	Nov 17, 2023 19:52	Address missing paragraphs from Word copy/paste issues
v. 1	Nov 17, 2023 19:47	Initial transfer from Word document