# Implementing Safety for Geneva actions

Contents

[Implementing Safety for Geneva actions 1](#_Toc192007463)

[Overview 1](#_Toc192007464)

[Pre-requisite reading 2](#_Toc192007465)

[Safety pre-check 2](#_Toc192007466)

[Health validation 3](#_Toc192007467)

[Error handling (‘rollback’) 3](#_Toc192007468)

[Organizing as ISafeOperation 3](#_Toc192007469)

[Risk Level self-assessment 5](#_Toc192007470)

[Touch Type 6](#_Toc192007471)

[Change Category 6](#_Toc192007472)

[SDP & Continuous validation 7](#_Toc192007473)

[System verification telemetry loop 7](#_Toc192007474)

[Safety Certification 7](#_Toc192007475)

[Swagger/Open Api caveat 8](#_Toc192007476)

[Safety in new operations 8](#_Toc192007477)

## Overview

Geneva Actions is a secure and compliant platform for running diagnostics and mitigation operations on management APIs in production to troubleshoot and resolve live site issues.

Ensuring the safety of ad hoc diagnostic and mitigation operations conducted on the GA platform is essential, as these operations grant access to the management APIs of various mission-critical services. The current gaps and proposed investments are elaborated in [Harden Geneva Actions Platform Safety.docx](https://microsoft.sharepoint.com/:w:/t/Acisandrunners/ER9JGSP0cu5FsUj4jiQ2NBIBwkXMzw9hXLlMqp5wTGHkzA?e=IvIbWJ).

The goal of this document is to detail high level implementation proposals for the key investments below.

1. Safety pre-check: Prior to a high-risk operation being executed, operation authors would have a chance to verify that prerequisites from a safety perspective are met.
2. Health validation: Post operation execution, health checks can be performed and validated.
3. Error handling : Allow operation owners to clean up state on failure.
4. Risk classification: Allow operation owners to apply a granular risk assessment
5. Continuous validation: Ensuring that operations are continually validated by a suite of tests, so that behavior is predictable when an operation is triggered.
6. Feedback loop: Telemetry driven feedback loop to validate the risk assessment and keep operations current.

## Pre-requisite reading

1. Understanding what an operation is expected to implement [C# SDK Geneva Actions Reference | Geneva Monitoring Docs](https://eng.ms/docs/products/geneva/actions/howdoi/csharpsdkrefga#operations---acissmeoperation)
2. Understanding the difference between coded and [swagger operations](https://eng.ms/docs/products/geneva/actions/howdoi/gaextensionswagger).

## Safety pre-check

Before executing an operation, authors might want to verify that the environment is in a safe state. For instance, before deleting a resource, ensure there is no current activity on it. While some pre-checks are built into downstream service APIs (like change safety in ARM APIs), others may require custom checks.

Currently, operation authors can perform these checks in the operation method. However, we can provide an ***optional*** method to remind authors to implement the check and promote this practice for high-risk operations in the future.

/// <summary>

/// Optional method to perform a safety precheck before executing the operation

/// </summary>

/// <param name="endpoint">Endpoint on which operation is invoked</param>

/// <param name="context">Execution context including parameters</param>

/// <returns>Operation response object indicating success or failure.</returns>

public virtual AcisSMEOperationResponse PerformSafetyPreCheck(IAcisSMEEndpoint endpoint, OperationExecutionContext context)

Note: Since only coded operations have an interface to expose methods, safety precheck will only be applicable to coded operations and not swagger/openapi.

## Health validation

Operations will be able to validate the health of a resource/s after execution. In order to do so, the operation author needs to specify the MDM account information and the health configuration.

From an api perspective, we will expose the following ***optional*** method.

Signature:

/// <summary>

/// Sets the health check settings for this operation using an overridable method.

/// </summary>

public virtual IEnumerable<ActionsHealthCheckConfiguration> SetHealthCheckConfigurationValues(IAcisSMEEndpoint endpoint, OperationExecutionContext context)

This work is already implemented and detailed in [Spec Health Integration.docx](https://microsoft.sharepoint.com/:w:/t/Acisandrunners/EXgjokTRP1VKn7cN4jRow_0BjXj-CEy6Q97NJ7nLO5kl-w?e=HX6udh)

Note: Since only coded operations have an interface to expose methods, health validation will only be applicable to coded operations and not swagger/openapi.

## Error handling (‘rollback’)

Since each operation in Geneva actions can perform multiple downstream state changes, a true rollback (like in the case of Ev2 where there is a last known good build to roll back to, or in the case of ARM where there is a previous template to reapply) is impractical. The closest we can offer is a standard method to handle errors. Operation authors can then attempt to clean up state. Upon completion, specific results will be returned to indicate whether cleanup was successful or not.

/// <summary>

/// Optional method to invoke cleanup actions after an operation fails.

/// This method should not assume the state in which the resource is left.

/// This is the closest to a rollback that can be done.

/// </summary>

public virtual IAcisSMEOperationResponse OnError(IAcisSMEEndpoint endpoint, OperationExecutionContext context, AcisSMEOperationResponse originalError)

Note : This is also only applicable to coded operations

## Organizing as ISafeOperation

All Geneva actions operations must implement an abstract class called AcisSMEOperation. We cannot require additional interfaces to be implemented by the operation author, or the currently working builds across the 500+ services onboarded to Geneva actions would break.

Purely for the sake of organization, we can however put the safety related primitives into an interface that IAcisSMEOperation then inherits from.

public interface IAcisSMEOperation : ISafeOperation

Proposed methods in ISafeOperation are

public interface ISafeOperation

{

    IEnumerable<ActionsHealthCheckConfiguration> SetHealthCheckConfigurationValues(IAcisSMEEndpoint endpoint,OperationExecutionContext context);

    IAcisSMEOperationResponse PerformSafetyPreCheck(IAcisSMEEndpoint endpoint,

       OperationExecutionContext context);

    IAcisSMEOperationResponse OnError(IAcisSMEEndpoint endpoint, OperationExecutionContext context);

}

The sequence of invocation is as follows

A diagram of a work flow

AI-generated content may be incorrect.

## Risk Level self-assessment

[Production Touch Safety.docx](https://microsoft.sharepoint.com/:w:/r/teams/VeenasNinjas/Shared%20Documents/General/OnePagers/Production%20Touch%20Safety.docx?d=w12038199d7eb46e18bbce7463b8c1459&csf=1&web=1&e=xXZ7xu) captures a granular classification of risk. Operation authors would be asked to self-assess their operations using this classification.

An enumeration would be introduced into the Acis SDK to represent the risk classification levels. Note: It is key that the risk classification levels are reviewed and finalized prior to implementation, since changing these in the future would be breaking for Geneva actions customers.

/// <summary>

    /// Classification levels for operation risk classification based on

    /// https://microsoft.sharepoint.com/:w:/r/teams/VeenasNinjas/Shared%20Documents/General/OnePagers/Production%20Touch%20Safety.docx?d=w12038199d7eb46e18bbce7463b8c1459&csf=1&web=1&e=2vrOZi

    /// </summary>

    public enum OperationRiskClassificationLevel

    {

        Low,

        Medium,

        High,

        Extreme

    }

A property would then be introduced to the AcisSMEOperation contract to represent the risk level.

/// <summary>

/// Specifies the operation risk classification level

/// </summary>  
public virtual OperationRiskClassificationLevel OperationRiskClassificationLevel => OperationRiskClassificationLevel.Low;

Note: We currently have a IsHighRiskOperation attribute in the interface, which would need to be deprecated after introducing the granular risk classification

## Touch Type

Currently Geneva actions only distinguishes between read and read-write for touches. To better categorize the type of touch we will add a new enum.

[Flags]

   public enum TouchType

   {

       Create = 0, // Adding a new item

       Read = 1, //Reading its value

       Update = 2, // Updating an existing item – changing its value

       Delete = 4, // Deleting an item

   }

An operation can classify itself as doing multiple types of touches.

## Change Category

Geneva actions operations currently are invoked in a wide variety of scenarios. To gather better insights about the scenario, a new enum will be introduced

public enum OperationCategory

    {

        Configuration,

        Data,

        Infrastructure,

        Application,

        Business,

        Security

    }

A more detailed description of each category is provided in [Geneva Actions Config Changes Risk Management.docx](https://microsoft.sharepoint.com/:w:/r/teams/WAG/EngSys/LiveSiteManagement/_layouts/15/Doc.aspx?sourcedoc=%7B4072598C-F060-4501-954C-4BF033650691%7D&file=Geneva%20Actions%20Config%20Changes%20Risk%20Management.docx&wdLOR=cFE2314C0-B1D9-46B1-8C40-94B021C3B9AC&action=default&mobileredirect=true)

## SDP & Continuous validation

[SDP is a feature](SDP%20for%20Package%20Publishing.docx) that was implemented last year in Geneva actions platform, to allow extension authors to specify a test suite that could be used to validate operations in the extension, for the purpose of promoting the package through the deployment rings. Extension authors are responsible for defining meaningful tests that validate the operations.

The test suite is currently run upon publishing. An [scheduled mode](https://msazure.visualstudio.com/One/_workitems/edit/31222487/) is currently being implemented.

## System verification telemetry loop

User attributed fields are prone to misclassification errors. Also, an operation may change over time and user attributed fields may become stale and need to be updated. A system verification mechanism is needed to keep risk classifications as current as possible. However, owing to limitations in availability and accuracy of data, such verification will always be best case effort and the accountability for accurate classification should rest with extension authors.

One stream of work undertaken is to statically analyze the intent of the operation against the declared metadata such as the “readwrite” flag or the operation name. This work will help in correcting

Dev spec: [Operation Intent Analysis.docx](https://microsoft.sharepoint.com/:w:/t/Acisandrunners/EeRya1v3vMpKo4WbAeLKjzgBb7z4pHhael72CWqD8ebVlA?e=L3AXI7&xsdata=%3D%3D&sdata=Q0FPWFdXWUZTUzloQjhvK1JQVEZ5ZWpla0V6MU0wbU5sYU5GZ2Jsc1puZz0%3D&ovuser=72f988bf-86f1-41af-91ab-2d7cd011db47%2Cbipinkr%40microsoft.com)

Another stream of work is underway to categorize the risk of the operation based on telemetry emitted by the operations, which can then be used to validate the risk classification level.

Dev spec: [Enhancing GA Operation Visibility and Risk Analysis.docx](https://microsoft.sharepoint.com/:w:/r/teams/WAG/EngSys/LiveSiteManagement/Shared%20Documents/Specs/Dev%20Specs/CTS%20Integration/2024-Se/Enhancing%20GA%20Operation%20Visibility%20and%20Risk%20Analysis.docx?d=wdf6fb689db734ca1b5bd854a09d13124&csf=1&web=1&e=1DRGJR&xsdata=MDV8MDJ8QmlwaW4uS3Jpc2huYW5AbWljcm9zb2Z0LmNvbXxmNWQ3NTc1Y2M5ZTc0MzIyMzZjZjA4ZGQzZmU0NDhhNHw3MmY5ODhiZjg2ZjE0MWFmOTFhYjJkN2NkMDExZGI0N3wxfDB8NjM4NzM2OTcxODgzMDIwNDEzfFVua25vd258VFdGcGJHWnNiM2Q4ZXlKRmJYQjBlVTFoY0draU9uUnlkV1VzSWxZaU9pSXdMakF1TURBd01DSXNJbEFpT2lKWGFXNHpNaUlzSWtGT0lqb2lUV0ZwYkNJc0lsZFVJam95ZlE9PXwwfHx8&sdata=V25FekRVWDlVbE5xcmY0ZWdUOXZYYWJRcUZOMGZ1U0RTd3M5TW9aOWpHMD0%3d)

## Safety Certification

The [Production Touch Safety.docx](https://microsoft.sharepoint.com/:w:/r/teams/VeenasNinjas/Shared%20Documents/General/OnePagers/Production%20Touch%20Safety.docx?d=w12038199d7eb46e18bbce7463b8c1459&csf=1&web=1&e=xXZ7xu) enumerates the guidance for minimum safety requirements for a production touch. Since an operation is the means to achieve the production touch, a validation process is needed to ensure that each operation meets those requirements.

The following sources could be used to validate the safety requirements.

1. Operational safety checks: By static analysis on operation code whether each operation implements if applicable, the members of ISafeOperation including safety prechecks and health integration.
2. Periodic functional tests: By telemetry analysis of an operation, whether operation has a suite of tests periodically running successfully.

A forum similar to the SFI forum would be responsible for reviewing the certification data of operations to mark them as safe. The forum can be aided by reports produced from the analysis and telemetry above, but can also override that based on data presented by the reviewing team.

If the forum is convinced about the safety of the operations, it can mark the operation as safe in the Geneva actions metadata. To facilitate that, we can stand up a Geneva actions management operation that would allow the metadata to be added/retrieved from Geneva actions storage.

## Swagger/Open Api caveat

Geneva actions operation authors have 2 ways in which operations can be specified. Coded operations (aka C# operations) and swagger/open api operations. Swagger operations are specified by providing an OpenApi specification, but no code. Hence the approach for safety prechecks, health integration, or error handling which defines standard methods in the code will not apply.

The best approach for incorporating safety checks, health integration into swagger, would be to invoke the swagger operation from within a workflow, in which case jobs for safety checks or health integration can be attached to the flow. Workflows would allow multiple swagger operations to be chained. (\*this feature is to be implemented) At that point, separate swagger apis for pre-safety check, health checks can be chained together.

## Safety in new operations

Geneva actions is introducing a new feature called Geneva actions workflows which allows multiple jobs to be chained together. For any new operation, we highly encourage extension authors to author their scenario in workflows where many of these safety constructs are easier to implement, than in the existing operations SDK.