package com.twitter.product\_mixer.core.pipeline.pipeline\_failure

/\*\*

\* Failures are grouped into categories based on which party is 'responsible' for the issue. This

\* is important for generating accurate SLOs and ensuring that the correct team is alerted.

\*/

sealed trait PipelineFailureCategory {

val categoryName: String

val failureName: String

}

/\*\*

\* Client Failures are failures where the client is deemed responsible for the issue. Such as by

\* issuing an invalid request or not having the right permissions.

\*

\* A failure might belong in this category if it relates to behaviour on the client and is not

\* actionable by the team which owns the product.

\*/

trait ClientFailure extends PipelineFailureCategory {

override val categoryName: String = "ClientFailure"

}

/\*\*

\* The requested product is disabled so the request cannot be served.

\*/

case object ProductDisabled extends ClientFailure {

override val failureName: String = "ProductDisabled"

}

/\*\*

\* The request was deemed invalid by this or a backing service.

\*/

case object BadRequest extends ClientFailure {

override val failureName: String = "BadRequest"

}

/\*\*

\* Credentials proving the identity of the caller were missing, not trusted, or expired.

\* For example, an auth cookie might be expired and in need of refreshing.

\*

\* Do not confuse this with Authorization, where the credentials are believed but not allowed to perform the operation.

\*/

case object Authentication extends ClientFailure {

override val failureName: String = "Authentication"

}

/\*\*

\* The operation was forbidden (often, but not always, by a Strato access control policy).

\*

\* Do not confuse this with Authentication, where the given credentials were missing, not trusted, or expired.

\*/

case object Unauthorized extends ClientFailure {

override val failureName: String = "Unauthorized"

}

/\*\*

\* The operation returned a Not Found response.

\*/

case object NotFound extends ClientFailure {

override val failureName: String = "NotFound"

}

/\*\*

\* An invalid input is included in a cursor field.

\*/

case object MalformedCursor extends ClientFailure {

override val failureName: String = "MalformedCursor"

}

/\*\*

\* The operation did not succeed due to a closed gate

\*/

case object ClosedGate extends ClientFailure {

override val failureName: String = "ClosedGate"

}

/\*\*

\* Server Failures are failures for which the owner of the product is responsible. Typically this

\* means the request was valid but an issue within Product Mixer or a dependent service prevented

\* it from succeeding.

\*

\* Server Failures contribute to the success rate SLO for the product.

\*/

trait ServerFailure extends PipelineFailureCategory {

override val categoryName: String = "ServerFailure"

}

/\*\*

\* Unclassified failures occur when product code throws an exception that Product Mixer does not

\* know how to classify.

\*

\* They can be used in failOpen policies, etc - but it's always preferred to instead add additional

\* classification logic and to keep Unclassified failures at 0.

\*/

case object UncategorizedServerFailure extends ServerFailure {

override val failureName: String = "UncategorizedServerFailure"

}

/\*\*

\* A hydrator or transformer returned a misconfigured feature map, this indicates a customer

\* configuration error. The owner of the component should make sure the hydrator always returns a

\* [[FeatureMap]] with the all features defined in the hydrator also set in the map, it should not have

\* any unregistered features nor should registered features be absent.

\*/

case object MisconfiguredFeatureMapFailure extends ServerFailure {

override val failureName: String = "MisconfiguredFeatureMapFailure"

}

/\*\*

\* A PipelineSelector returned an invalid ComponentIdentifier.

\*

\* A pipeline selector should choose the identifier of a pipeline that is contained by the 'pipelines'

\* sequence of the ProductPipelineConfig.

\*/

case object InvalidPipelineSelected extends ServerFailure {

override val failureName: String = "InvalidPipelineSelected"

}

/\*\*

\* Failures that occur when product code reaches an unexpected or otherwise illegal state.

\*/

case object IllegalStateFailure extends ServerFailure {

override val failureName: String = "IllegalStateFailure"

}

/\*\*

\* An unexpected candidate was returned in a candidate source that was unable to be transformed.

\*/

case object UnexpectedCandidateResult extends ServerFailure {

override val failureName: String = "UnexpectedCandidateResult"

}

/\*\*

\* An unexpected Candidate was returned in a marshaller

\*/

case object UnexpectedCandidateInMarshaller extends ServerFailure {

override val failureName: String = "UnexpectedCandidateInMarshaller"

}

/\*\*

\* Pipeline execution failed due to an incorrectly configured quality factor (e.g, accessing

\* quality factor state for a pipeline that does not have quality factor configured)

\*/

case object MisconfiguredQualityFactor extends ServerFailure {

override val failureName: String = "MisconfiguredQualityFactor"

}

/\*\*

\* Pipeline execution failed due to an incorrectly configured decorator (e.g, decorator

\* returned the wrong type or tried to decorate an already decorated candidate)

\*/

case object MisconfiguredDecorator extends ServerFailure {

override val failureName: String = "MisconfiguredDecorator"

}

/\*\*

\* Candidate Source Pipeline execution failed due to a timeout.

\*/

case object CandidateSourceTimeout extends ServerFailure {

override val failureName: String = "CandidateSourceTimeout"

}

/\*\*

\* Platform Failures are issues in the core Product Mixer logic itself which prevent a pipeline from

\* properly executing. These failures are the responsibility of the Product Mixer team.

\*/

trait PlatformFailure extends PipelineFailureCategory {

override val categoryName: String = "PlatformFailure"

}

/\*\*

\* Pipeline execution failed due to an unexpected error in Product Mixer.

\*

\* ExecutionFailed indicates a bug with the core Product Mixer execution logic rather than with a

\* specific product. For example, a bug in PipelineBuilder leading to us returning a

\* ProductPipelineResult that neither succeeded nor failed.

\*/

case object ExecutionFailed extends PlatformFailure {

override val failureName: String = "ExecutionFailed"

}

/\*\*

\* Pipeline execution failed due to a feature hydration failure.

\*

\* FeatureHydrationFailed indicates that the underlying hydration for a feature defined in a hydrator

\* failed (e.g, typically from a RPC call failing).

\*/

case object FeatureHydrationFailed extends PlatformFailure {

override val failureName: String = "FeatureHydrationFailed"

}