Tranzact API Service Level Agreement

Tranzact's API versioning is aligned with standard **semantic versioning**. This enables both external and internal customers to have a clear record and expectations associated with changes based on their impact. The change management plan ensures consistent and clear communication with each consumer of the API, preventing unplanned complications on the client-side development.

Semantic versioning: consists of breaking down changes in development in three distinct positions within the version nomenclature

Version 1.1.2
Version + Major.Minor.Patch

Given a version number MAJOR.MINOR.PATCH, the following will increment:

1. *MAJOR version for incompatible/breaking API changes**
2. *MINOR version for additional functionality in a backwards-compatible/non-breaking manner*
3. *PATCH version for backwards-compatible/non-breaking bug fixes.*

**See Definitions at the end of this section*

Patch

Impact: A patch should have no impact on customers. All patches should be backwards compatible as they are only fixing incorrect behavior.

Required Development: None

Timeframe: N/A

Communication:

- Patches will only be communicated if they include fixes that are visible or affect a customer in some manner. We may communicate on a customer specific basis in this case.

Example:

Client A is on V 1.0.0

01/10: Client or Internal development team reports unexpected behavior.
01/12: Patch is released to correct behavior and customers notified IF patch was customer impacting. API Version increments to 1.0.1.

*Dates between report and release are an example and not a commitment. LOE on implementing a patch can vary.

Minor

Impact: Minor Updates are considered backwards compatible *assuming best practices*. These changes are additions of non-required properties. A potential scenario for API downtime in clients would be the client side development having a hard coded response structure that does not admit an addition of new attributes.

Required Development: Optional. Based on client side development and requirements. A customer may want to consume the additional feature, in which they would need to potentially develop against it to return it to users. Level Of Effort (LOE) would be Low.

Timeframe: N/A

Communication:

- 3 Weeks Before Release:
  - You should expect an email with the expected changes
  - The Release Calendar will be updated with current Release/End of Life Dates
- On Release
  - Detailed Release Wiki that walks you through changes--why we made them how to take advantage of them (if you want)

Example:

Client A is on V 1.0.0

01/07: Client receives notice of upcoming minor changes.
01/28: Minor change released, API version incremented to 1.1.0. Release wiki shared and client side development to consume is optional with a Low LOE.

Major

Impact: A major release by definition involves a breaking change. This will result in the eventual need for customer development in order to transition to latest version.

Required Development: Yes. LOE will vary from version to version. Even best practice development is affected.

Timeframe:

- Maximum update frequency: once every 12 months.
- Announcement Date: About 6 weeks before release date.
- Minimum transition time: 1-12 months from release date to develop against breaking changes. After 12 months, previous version is at risk of being deprecated.

Communication:

- 6 Weeks Before Release:
  - You should expect an email with the expected changes
  - The Release Calendar will be updated with current Release/End of Life Dates
- On Release
  - Detailed Release Wiki that walks you through changes--why we made them how to adjust to them
- On Previous Version Deprecation
  - An email will be sent three weeks before a version is deprecated to customers still on that version to inform them that we will no longer be able to provide support per our policy.

Example:

Client A is on V1.

April 2020: V2 Announcement broadcasted to partners, scheduled for July 2020. Release wiki provided for LOE assessment and sprint planning, subject to minor changes.

July 2020: V2 Released on time, all developers have received Release Wiki and now have a final updated API Wiki. 12 month timeline starts for transition. Both API versions are supported.

July 2021: V1 support stops and is in line to be deprecated in the upcoming months*.

December 2023: V3 announcement date. V2 has been deprecated at this point.

*V1 deprecation date is guaranteed to be after 12 months of V2 release, there is no set time frame on deprecation date after support stops at the 12 month mark. Notice of version being deprecated will be provided with 6 weeks in advance.

Definitions

- **Code change:** A restructuring of the request or response JSON format. It is considered "breaking" (MAJOR version) when attributes are removed or existing structure is modified. *See Incompatible/breaking change*
- **Configuration change:** The addition of a non-required property, attribute or value option. It is considered "non-breaking" (PATCH OR MINOR). *See Backwards compatible/non-breaking change*
- **Bug Fix:** an internal change that fixes incorrect behavior.
- **Incompatible/breaking change:** change that requires development on the client side to continue running the API. IF deployed to PROD without the customer developing on the client-side to accomodate for the change THEN customer's original development will break as soon as the change is pushed to PROD. Some of these changes would be:
    - Code changes
- **Backwards compatible/non-breaking change:** change that should not require development on the client side to continue running the API. IF deployed to PROD without the customer developing on the client-side to accomodate for the change THEN customer's original development still isn't affected. Some of these changes would be:
    - Configuration changes
        - Addition of property or attribute in response
        - Addition of non-required property or attribute in request
    - Bug fix

Support

Only 2 versions will be supported in parallel at any point in time. This enables the best customer support possible. Customers will always have at least 12 months to develop against latest version based on the communication plan outlined in each of the update type strategies above.

Resources

API Wiki: the API wiki will be updated for every minor and major release.
Release wiki: will be provided for appropriate review of changes pushed to production in minor and major releases.
Release calendar: will be kept up to date with all relevant dates for upcoming planned releases. (Merge with release notes OR add to Zayo.com website?)