TIPS Live change of the system
Releasing new versions when the system is running

TIPS Contact Group #10
**TIPS change and release**
What the application does for making possible to deploy versions without disruption of the service.

**Case studies**
Some examples about how this will presumably happen in some cases on the horizon.

**Wrap-up**
Just a short recap.
TIPS Change and release

How the change & release approach works
TIPS change and release

✗ TIPS compatibility mode
✗ Canary release process

In a nutshell:
TIPS is operated h24, mostly unmanned. This is possible because critical parts of the system has been designed as clusters of redundant nodes. Installing new versions of software in a H24 system requires special care: ad hoc designed application features shall prevent service disruptions and reduce the risk of malfunctioning. Nonetheless, each change must be designed having in mind, since the beginning, how it will be installed into production.
As seen before, TIPS has two types of redundant nodes:

- **Sequencing**
  - "horizontally scalable"

- **Parallelization**
  - "vertically scalable"

**Types of node**

- **ACTIVE**
  - Hot-ready
  - Hot-ready
When a “parallel” node fails, the others take the entire workload.

For “sequential” nodes, next eligible node becomes active.
24/7/365 means “no maintenance window”

How to change software version when service is running?
- Maintenance, Infrastructure, Patch management, etc.
- System configuration Deployment of new versions, configuration data injection, etc.
e.g.: new configured instance must be really active only after some checks of correctness have been done

«Not eligible messages»: messages are not used for output (used for comparison)
Compatibility mode

To allow continuous operation:

- Any change in the system should be designed to be deployed in **two steps:**

  - **Compatibility mode** when new and old behaviors can coexist

  - **New-function mode** when old behavior is not allowed anymore
Canary deployment

It is a way to control and minimize impacts of deploying new versions into production.

**First**
A slow rolling out of the change involving a small number of payments (marked as yellow messages)

**After**
Rolling out to the entire platform
As seen before, TIPS has two types of redundant nodes:

Parallel components: «Canary deployment»

Sequential components: «compatibility mode»
Case study

How the **change & release** approach applies to some changes
TIPS C&R

TIPS 0010 - sending of reports to multiple DNs

- Change must be backward compatible
- Not suitable for compatibility mode (only horizontally scalable nodes are affected)

- Two steps:
  - First, all TIPS nodes will adopt new datamodel - here canary mode will be used
  - Second, new datamodel can be filled using CRDM (deploy of CRDM change during MW).

New function is available to the community of users
TIPS 0008 - Reset of CMB headrooms – 1/3

- Change must be backward compatible
- Both **canary mode** and **compatibility mode** will be used
TIPS 0008 – Reset of CMB headrooms – 2/3

Four steps:
- First, “horizontal” nodes will be upgraded to new version using **canary mode** (new function will not be processed because schema validation blocks ModifyLimit with HDRM codeword)
- Second, “vertical node” will be deployed using **compatibility mode** – still dormant new function
TIPS C&R

Examples

TIPS 0008 – Reset of CMB headrooms – 3/3

- **Four steps:**
  - **Third**, XML schema is updated to accept new message format – change can be deployed in **canary mode**
  - **Fourth**, after a few checks, canary mode is switched off

**New function is available to the community of users**
Case one: a new mandatory tag is added to a schema, from $T_f$.

**Compatibility mode 1**
- From $T_0$ all community must be able to accept the new tag (maybe ignoring it) as optional (e.g.: 2 versions of message).

**Compatibility mode 2**
- From $T_1 > T_0$ senders can produce messages with new tag.

**New function mode**
- From $T_f > T_1$ all senders must produce messages with the new tag. - Change is complete
Case two: a new **optional** tag is **added** to a schema, valid from $T_f$

**Compatibility mode**
- From $T_0$ all community must be able to accept (maybe ignoring it) the new tag

**New function mode**
- From $T_f > T_0$ senders can produce messages with new tag

*Change is complete*
Case three: a **mandatory** tag is **dropped**, from $T_f$

**Compatibility mode**
- From $T_0$ the tag is made **optional** (e.g.: accepting 2 versions of the message)

**New function mode**
- From $T_f > T_0$ all senders are **not allowed** to produce messages with the dropped tag anymore

**Change is complete**
Case four: an **optional** tag is **dropped**, from $T_f$

**New function mode**

- From $T_f$ all senders **are not allowed** to produce messages with the dropped tag anymore

**Change is complete**
Wrap-up
Time for a recap
Recap

- TIPS operates 24/7/365
- New features
- Compatibility mode
- Canary mode
- New processes
- Backward compatibility / coexistence
- System must be updated progressively
- Tailored and designed together with the change
- TIPS is a distributed system
THANK YOU!

Any questions?