Infosys | Finacle

FINACLE SUPPORT CONNECT

Knowledge Series

August 2021 | Volume 32

We are glad to announce that **Purna Chander Rao P** has taken over as the new **Head- Finacle Global Support**. Purna has been an integral member for the Support Connect Knowledge Series panel since its inception. He has been a part of Infosys and Finacle for over 20 years - catering to new service offerings and product lines and helping clients achieve their business objectives. He has an extensive Finacle experience of working with banks around the globe.

We are confident that his many years of Support experience and expertise will strengthen us to bring more useful articles before you. Our endeavor to bring you a knowledge bulletin with best practices and resolutions is only strengthened with Purna's new role as Head of Finacle Support. We hope you join us in congratulating him as he begins his new journey.

Purna Chander Rao P
Head- Finacle Global Support

In this edition you will find the following articles:

- **LOCK and RETRY Parameters for Transaction Posting**
- **Holiday Configurations in Finacle Online Banking**
- **Thread Dump Analysis in WAS**

So let's start reading!

LOCK and RETRY Parameters for Transaction Posting

Product: Finacle Core Banking Version: 10.x onwards

During transaction posting, the system gets a lock on the **GAM** table record of the entities and post the transactions. Once the transaction is posted, the respective **GAM** table record is updated, and the lock is released.

Resource Busy error is observed for the concurrent transactions hitting the same office or customer accounts, while the transactions are left in the entered status. Such cases can be avoided by configuring the two environment variables given below:

INTV_LOCK_POST: Indicates the interval that the system must wait and acquire the lock on the entities. The default value is 500000 micro sec i.e., 0.5 seconds.

POST_LOCK_RETRY: Indicates number of times that the system will attempt to acquire the lock and post the transaction. The default value is 60 attempts.

The parameters in the different versions are as follows:

10x: **INTV_LOCK_POST and POST_LOCK_RETRY**

11x: **INTV_LOCK_POST and DFLT_LOCK_POST_TRY_CNT**

By default, the system will wait for 0.5 seconds to acquire the lock and attempt to post the transaction. This loop will continue for 60 times i.e., the system will keep trying to acquire the lock and post the transaction for a time period of 30 seconds. This time period can be modified as per requirement for different scenarios.

This can be used for various Finacle Services where **LISRVR** is being used. Configuring it in **commonenv.com**, it becomes applicable for all types of transactions. It can also be made service-specific by configuring in the respective service start script.

For example, adding these variables in the **swiftsrv** start script will make it applicable for **SWIFT** transactions.

Another example is that in C24, we can set these variables in **Start-uniser** or **uniser.cfg** files. We can set this env variable value as **INTV_LOCK_POST=250000**. [i.e. 0.25 sec]. By reducing the value of this env variable [from the default value of 0.5 seconds to 0.25 seconds], we can reduce the time period for which the system waits to acquire the account lock, thus increasing the probability of a **lisrvr** instance to acquire the lock on the account. This would help us to minimize the time delay to an extent, thereby reducing the instances of **SIGALRM**s.

For more information on this, please connect with the Support team.

Holiday Configurations in Finacle Online Banking

Product: Finacle Online Banking Version: 11.0.x onwards

The **Bank Holiday** functionality can be enabled in Finacle Online Banking application to restrict users from initiating hot or scheduled transactions, transactions posted via specific networks, and execution of batches. In case of recurring transactions, an option is provided to users during transaction initiation to select a **validity indicator**, if the transaction falls on the bank or network holiday.

- Bank holidays can be enabled or disabled through the **Property Manager** table**,** PRPM parameter **BANK_HOLIDAY_ENABLE**
- Bank admins can create multiple holiday calendars via **Application Maintenance → Maintain Bank Holiday → Create New**
- Whenever there is a change in the holiday list, the bank has to refresh the cache ID **BANKHOLIDAYLISTCACHE** and **restart** the application for changes to take effect
- Admin has to map one of the holiday calendars as the bank holiday in **PRPM** table against the property name **BANK_HOLIDAY_CAL_TYPE**
- In case, holidays need to be defined specific to the networks, then the corresponding calendar should be mapped against the required networks on **NPRT** table via **Transaction Administration→ Financial Transaction Maintenance→ Network Properties**
- Validity indicator for:
    - Recurring transactions will be stored in **TRQH.VALIDITY_INDICATOR** (**N** – NEXT; **S** – SKIP; **P** – PREVIOUS)
    - Recurring scheduled FEBA batches will be stored in **SCDT.HOLIDAY_OPTION**, with one of the below values:
        - **NA** - Holidays will not be considered. Task gets activated on holidays as well
        - **YS** - Holidays will be considered and Task activation is skipped if it falls on holiday
        - **YN** - Holidays will be considered and Task activation is postponed to next day same time
        - **YP** - Holidays will be considered and Task activation is preponed to previous day same time
- Ensure the holidays maintained for online banking transactions are in sync with the back-end core application

The tables that are linked with the functionalities are **HOLT**, **PRPM**, **NRPT**, **TRQH**, and **SCDT**.

Thread Dump Analysis in WAS

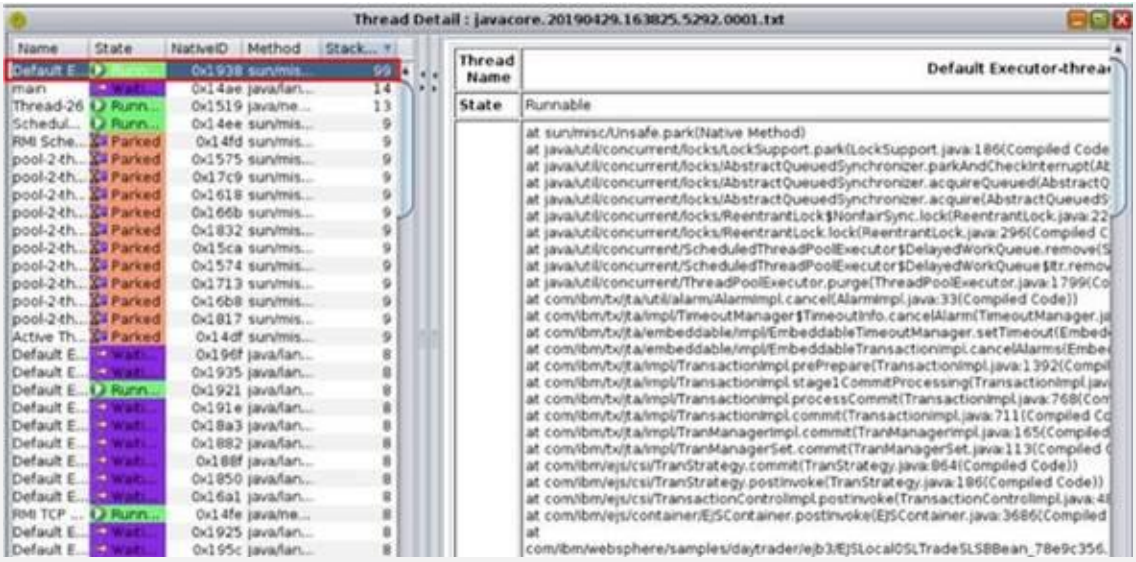
Product: Webserver Application Server (WAS)

Thread dump analysis plays a significant role in debugging any issues related to application slowness. A thread dump is a snapshot of the state of all threads that are part of the process. The state of each thread is presented with a stack trace, which shows the contents of a thread's stack. Some of the threads belong to the Java application running, while others are **JVM** internal threads. **IBM Thread and Monitor Dump Analyzer for Java (TMDA)** is tool that allows identification of hangs, deadlocks, resource contention, and bottlenecks in Java thread dumps. Thread dumps can be generated running the command, **kill -3 ${WAS PID}** in the server, where the resource contention is observed.

It is recommended to always take 6-7 thread dumps with a time interval of 10 seconds each and then analyze and consolidate the findings for a clear picture.

For any clarifications, please reach out to the Support team.

The image below is an example of thread dumps generated.



Hope you like this edition. Is there anything that you'd like to see in the forthcoming series? We'd love to hear from you!
Write to us at finaclesupport@edgeverve.com