# 

# https://cloud.vn.dektech.internal/index.php/s/Rn3dUoYJNCLUuVe

# 

**Client**

ILES – Communicator

Install node, have job ITRSender to send ITR

Eiffel core 🡪 create queue. Find address message bus.

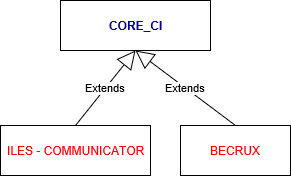
Config for Iles – Communicator and Eiffel core

RESTART, START: trigger sync log

**MANAGER**

WAITTING, REJECT: stop sync log

ENDLOOP: finish test, Query BTF Endloop will received BTF



**Iles – communicator: (Client)**

**(Fem036)**

**Global config**

* Eiffel Message Bus
* Host name
* Exchange name
* User name
* Password
* Component name
* DomainId
* Queue length: 100

**In MANAGER**

* Eiffel rule trigger

**Build**

* **C1:** Execute system Groovy Script
* Create queue
* Display BTF event
* Save BTF event to eventDao
* Show badge info
* **C2:** Manage BTF Event: ManageBTFBuildStep.java // Receive BTF events with specific tag and binding keys then save to event DAO.
* Tag (fem)
* Binding keys
* Event DAO 🡪 path to store BTF event
* Event Queue 🡪 queue name to store BTF event
* Use HashSet to remove all duplicate job name.
* Create queue name ( EiffelEventReceiver)
* Start receiver ( sleep 5000 mili)
* Stop receiver
* Count number BTF event type
* Dislay BTF event
* Trigger requester job if RESTART or store to event DAO
* Check jon name exist
* Add value to btfScore
* Save BTF event to EventDao
* Show badge info
* Check trigger job not exist
* BTF Receiver: 🡪 BTFReceiverBuildStep.java
* Perform
* List secondaryBinding
* List Timestamped BTFEvent
* Show BTF in the end log in vise

**IN VISE 0203**

* Parameter:
* Boolean parameter: ILES TOOL, INSTALLATION, PROVISIONING, TESTEXEC, STORE\_PASSED\_PCAP,
* String parameter: Phoenix, Provisioning\_exec, test\_exec, config.properties, MTAS\_SW\_or\_BL, MTAS\_PDB, CSCF\_SW\_or\_BL, CSCF\_PDB, PCSCF\_SW\_or\_BL, PCSCF\_PDB, IBCF\_SW\_or\_BL, IBCF\_PDB, AGW\_SW\_or\_BL, AGW\_PDB, TRGW\_SW\_or\_BL, TRGW\_PDB, ARTIFACT, TESTCASE\_TAGS, JOB\_DESCRIPTION,
* Dynamic choice parameter: MTAS\_VERSION, CSCF\_VERSION, PCSCF\_VERSION, IBCF\_VERSION, AGW\_VERSION, TRGW\_VERSION, INT\_VERSION,
* Text parameter: EMAIL\_NOTIFICATION

**Config in Build**

* Execute groovy Script: add file groovy script
* Stand\_alone\_gui\_parser.groovy ???
* Create\_config\_properties\_file.groovy ???
* ITR sender: send ITR 🡪 ITRSenderBuildStep.java //Create and send ITR event.
* Node Type
* Node Version
* Loop Type
* Parameter: INSTALLATION\_ABLE, PROVISIONING\_ABLE, TESTEXEC\_ABLE, MTAS\_CONFIG, CSCF\_CONFIG, IBCF\_CONFIG, PCSCF\_CONFIG, INT\_CONFIG, TAG, VISE\_INFO, AGW\_CONFIG, TRGW\_CONFIG, ITR\_SENDER\_JOB\_NAME, ITR\_SENDER\_JOB\_BUILD, CLIENT\_JENKINS, PHOENIX\_TOOL\_PATH, PROVISIONING\_TOOL\_PATH, TESTEXEC\_TOOL\_PATH, QUICK\_TEST
* Get event value form Build
* Log will print in VISE
* Add ITR Event to build params
* Check execution condition
* Check param condition
* Query BTF: received BTF in BTF Queue 🡪 BTFEventQueryBuildStep.java //This build step will continuous query in a specific event DAO to find an BTF event matching condition. Then it will add the event to build parameters.
* Delay time: 2 🡪 3
* Waiting time: 10 🡪 120
* BTFEvent Received type: RESTART, STARTLOOP, WAITING, REJECT
* BTFEvent Rejected type: WAITTING, REJECT.
* The build parameter name: WORKER\_URL
* Event DAO path
* Event queue name
* Add buildLogURL 🡪 Build parameter action
* Print log BTFReceived in VISE
* Add received BTF event to build parameter action to debug
* Check if BTF event contain Reject
* Get event ID
* Query event with ID matching
* Remove all event is overtime 🡪 check the BTF event is over time out ( isExpireEvent, +3..only receive string before “UTC”, remove randomNumber to compareTo.

CurrentTime: BTFevent+String “**MM-dd-yyyy-HH:mm:ss-z” +UTC**

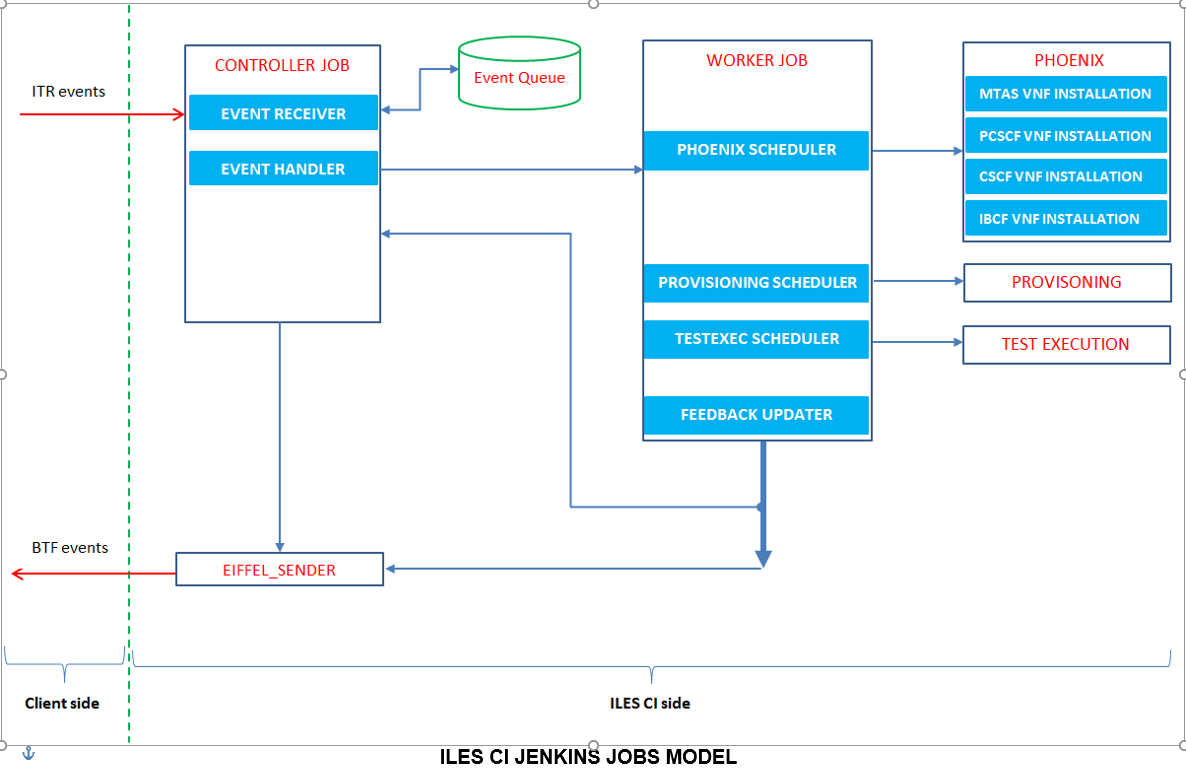
**BTFEventTime:** BTFevent+String “**MM-dd-yyyy-HH:mm:ss-z” +UTC + randomNumber**

* Add the parameter value used for a build
* Execute shell - Sync log shell script: will turn on in config execute shell
* BTF Query:
* Delay time
* Waiting time
* BTFEvent Received type: ENDLOOP
* BTFEvent Rejected type
* The build parameter name
* Event DAO path
* Event queue name

**Config in Post – Build Action**

* Editable Email Notification
* MakeBadgePostBuildStep: information will show bellow build number.
* MakeBadgePostBuildStep.java
* List BTFEventParameter
* List ITREventParameter
* Check if ITR not sent successfully
* Check if it doesn’t enough BTF event
* Add EventBadge (Reject, waiting, finish, testscore)
* Add badge to build history ( get/setDescription)
* Check this build was received enough BTF or not

Beccause At client will have 3 event ( ITR, BTF Startloop, BTF Endloop) will check 2 event BTF in ( isReceivedEnoughBTF), if only receive 1 event ( not Reject) return false.



**In Becrux: Fem38**

**In Global config**

* Config message bus
* Config secondary binding 🡪 SecondaryBinding.java
* Vise Channel Global Configuration
* Vise channel DAO path
* Vise Channel Pools
* ILES CI Configurations
* Iles guardians e-mail addresses
* ILES Tools/Script Directory path
* Component DAO path
* Event DAO path

**In CONTROLLER**

* Eiffel Rule Trigger – Binding key: the queue binding key for the job to listen and trigger, this job will be triggered automatically if a coming event have routing key match with this binding key.

**Build**

* Eiffel event receiver 🡪 IlesEiffelEventReceiverBuildStep.java // for receiving events
* Received ( tag, receivedTypes, secondaryBinding, timeout, customBinding)
* Set tag
* List received event type
* Add secondary to Globalconfig
* Set timeout 5000
* If tag not null, will received event
* Event Handler: Strategy for All Event 🡪 AllEventStrategyRegistrationBuildStep.java // for receiving events
* AllEventStrategy..( workerJobName, poolName, votingThreshold)
* Super( workerJobName, poolName) 🡪 ITREventStrategy… 🡪 EventStrategy..
* Set votingThreshold ( number test pass, will include in config )
* Event Handler 🡪 IlesEventHandlerBuildStep.java //Handle events follow event strategies
* IlesEventHandler..(maxRetries, maxTimeout, queueName)
* List of assigned Event Type
* Update maxRetries, maxTimeout
* Check existing event strategy
* Get all event received from Eiffel Receiver or upstream job
* List all received event
* Received event
* Load all event in DB
* Count total input event
* Start handling
* Set value to used = true
* Put all feedback event to EiffelEventParameterValue\_ToSend 🡪 Eiffel sender can send out
* Save event to DB (EventDao) to handle in next execution.
* Add brief description ( addEventBadge ( addAction ( yellow color), addDescription) will show below build number)
* ITREventStrategy
* Get vise channel form event Or get vise channel from pool . Check vise available
* Reserved vise channel
* Get installable, pdp, version, from parameter
* Get config properties path
* Get manual run params ( get BuildParams)
* List node for test, all node for test
* Prepare for test job
* Prepare for baseline job
* GetITRJSenderJobName
* getITRSenderjobBuild
* Get ITRJob requester
* Check loop running or voting
* Run loop
* Validate event
* Get tag from ITREvent
* Create feedback BTF
* Get version helper

**Post-Build Action**

* Trigger DownStream Job: EIFFEL\_SENDER // for sending BTF events with notification + errors if any.

**In WORKER**

**Build**

* MultiViseManager update identifier to current job 🡪 MultiViseManagerBuildStep\_UpdateIdentifer.java// update identifier to reserved VISE
* Phoenix installation Build Scheduler 🡪 PhoenixSchedulerBuildStep.java // process Installation step
* Perform
* Initialize an empty ResultParameterValue
* Add to NwftBuildParameter ( global parameter)
* List CommonParameterValue
* Update ResultParamaterValue if Skipped by user.
* Get the source folder of script
* Add to NWFTParameter
* List ComponentParameterValue
* List ReservedViseChannelParameterValue
* Create Map jobsMap?????
* List Action ( NwftBuildParameterAction)
* Add action( parameter) in 3 list ( Common, Component, Reserved)
* Add jobsMap
* Update ResultParameterValue
* Get link of build
* Return Result
* Provisioning Build Scheduler 🡪 ProvisioningSchedulerBuildStep.java// process provisioning step
* Perform
* Initialize an empty ResultParameterValue
* Add to NwftBuildParameter ( global parameter)
* Update ResultParamaterValue if Skipped by user.
* List Action ( NwftBuildParameterAction)
* List ReservedViseChannelParameterValue
* Add param NwftBuildParameterAction in list action
* Update ResultParameterValue
* Get link of build
* Return Result
* Test Execution Build Scheduler 🡪 TestExecutionSchedulerBuildStep.java// process testing step
* Perform
* Initialize an empty ResultParameterValue
* Add to NwftBuildParameter ( global parameter)
* List Action ( NwftBuildParameterAction)
* Update ResultParamaterValue if Skipped by user.
* List ComponentParameterValue
* List ReservedViseChannelParameterValue
* Add param NwftBuildParameterAction in list action
* Update ResultParameterValue
* Get link of build
* Get TestScore
* Return Result

**Post – Build Action**

* Loop feedback updater 🡪 LoopFeedbackUpdaterPostBuildStep.java// update the processing result, create BTF event and trigger Eiffel sender job to send event.
* Create/ update BTF
* Set product, set baseline, get vise channel used / set BTF vise channel, set phase, phaseStatus, set testcore, result, jonId, BTFJobId, …
* Correct BTF PhaseStatus and Phase
* Get Result log
* Get TestScore
* Get source folder
* Update baselineProcessing, change LoopRunning
* Save this BTF event to EiffelEventParameterValue\_ToSend so that it will be sent back to client
* Set tag in routing key
* Trigger Eiffel\_sender job to send BTF back
* Trigger DownStream Job 🡪 CONTROLLER // trigger CONTROLLER again

Processing Result:

[Name:PROVISIONING] [Status:SUCCESS] [Detail: (LINK:https://fem038-eiffel021.rnd.ki.sw.ericsson.se:8443/jenkins/job/PROVISIONING/202/) ] [Result: [Source Path: /proj/ims\_lu/cba\_cde/int\_ci/deliveries/provisioning/1.0.8/provision] ]

**In PHOENIX**

**Build**

* Use Phoenix to install node 🡪 PhoenixInstallationBuildStep.java// process phoenix script
* Perform
* List ComponentParameterValue
* List ReservedViseChannelParameterValue
* Get commandParameter( component, vise)

🡪If customversion will add “sw”, else not

**Post – Build Action**

* Flexible publish: // gathering log and save in to build artifacts

In **PROVISIONING**

* Perform provisioning 🡪 ProvisioningBuildStep.java
* Perform
* Get ProvisioningPropertiesFromBuild
* Validate Provisioning
* Execute Provisioning command

**In TESTEXE**

**Build**

* Execute INT test suite 🡪 TestExcuteBuildStep.java
* Perform
* Get config file form NwftParameterValue
* Check vise channel 🡪 getTestExecPropertiesFromBuild
* Add badge build
* Validate properties
* Execute shell command
* Get QUICK\_TEST parameter
* Execute system Groovy Script
* Execute shell

**Post - Build Action**

* Publish Junit test result report

**IN EIFFEL\_SENDER**

* Eiffel event sender 🡪 IlesEiffelEventSenderBuildStep.java

|  |  |  |
| --- | --- | --- |
|  | AllEventStrategyRegistrationBuildStep | Assign strategies for all Event types. |
|  | MultiViseManagerBuildStep\_UpdateIdentifier | Create a reservation identifier for chosen VISE channel. The VISE channel will get from ReservedViseChannelParameterValue in the build parameters.  In Worker |
|  | IlesEiffelEventReceiverBuildStep | The build step for receiving Eiffel events with condition. Create EiffelEventParameterValue\_Received build parameters for every received event. |
|  | IlesEiffelEventSenderBuildStep | This build step will get all EiffelEventParameterValue\_ToSend from build parameters and send. |
|  | PhoenixInstallationBuildStep | Process Installation using Phoenix script. The build step will find ComponentParameterValue in build parameters and process install that Component in VISE channel (defined in ReservedViseChannelParameterValue in build parameters) |
|  | PhoenixSchedulerBuildStep | Trigger phoenix jobs with condition (CommonParamenterValue & ComponentParamceterValue from build parameters).  *Job name*: name of the job for trigger Installation for every component  After finish, the build step will gather result and create ResultParameterValue and add to build parameters. |
|  | ProvisioningBuildStep | Perform Provisioning step. |
|  | ProvisioningSchedulerBuildStep | Trigger Provision follow condition in build parameters (CommonParamenterValue.Provisioning = ‘true’).  After finish, the build step will gather result and create ResultParameterValue and add to build parameters. |
|  | TestExecutionBuildStep | Run NWFT, all properties are optional support for testing process. |
|  | TestExecutionSchedulerBuildStep | Run Network Features Test follow condition in build parameters (CommonParamenterValue.Provisioning = ‘true’).  After finish, the build step will gather result and create ResultParameterValue and add to build parameters. |
|  | LoopFeedbacksUpdaterPostBuildStep | This build step updates all processing result of Phoenix, Provisioning and TestExec into a BTF event. Create an EiffelEventParameterValue\_ToSend for BTF event and trigger sender job for sending Eiffel event.  *Sender job name:* name of sender job. |

**Fem035**

**ITR GUI Delivery**

**Fem038**

**Simulated Standalone**

**Fem037**

**Simulated production**

**Fem036**

**Vise Gui**