Using TIBCO Fulfillment Orchestration Suite for Resource Order Management

About Behaim IT Solutions

Behaim IT Solutions partners with clients to help them overcome their 21st century IT challenges: Leveraging the Cloud and shared services; Gaining insight quickly in a world of Big Data; and taking real-time action on Data for competitive advantage, new opportunities, and risk diversion. We specialize in Integration and Hadoop technologies that enable: Data Storage, Data Analysis, Integration / Internet of Things (IOT), API Management, In-memory & Complex Event Processing, Business Process Management, Master Data Management, and Visualizations.

1. Executive summary

TIBCO Fulfilment Orchestration Suite (FOS) and TIBCO integration platform have a proven track record of implementing a modern and dynamic catalog-driven Service Order Management system. Based on numerous successful TIBCO FOS implementations performed by Behaim IT's consulting team, Behaim's Resource Order Management Accelerator for TIBCO FOS is complementing TIBCO FOS and TIBCO integration platform in the areas of project methodology, overall system setup, corner integration cases, and smart testing strategy for Resource Order Management, for the activation and other functional management of network elements.

	ABOUT THE AUDIENCE
TARGET AUDIENCE	Telco operators / mobile providers who: • Seek a better solution for ROM (Resource Order Management) and Provisioning • Consider TIBCO Fulfillment Orchestration Suite (FOS) for ROM in addition to FOS's typical role as a Service Order Management (SOM) system • Operate outdated resource provisioning platforms
CHALLENGES FACED	• Problems due to legacy integration from many years ago • Complex provisioning and activation flows • Obsolete protocols in legacy network components • High integration OpEx costs caused by intermediate integration platforms, e.g., InstantLink

ABOUT THE ACCELERATOR

ACCELERATOR OBJECTIVES	• Quickly accelerates FOS implementation • Reduces implementation risks
ACCELERATOR BENEFITS	• Leads customer through FOS implementation • Describes the reference architecture • Brings the proven methodology for the entire project lifecycle • Includes tools and components
ACCELERATOR CONTENT	• Methodology • Prepopulated document guides • Code templates • Value-added tools and components • ... and more

2. Motivation

Communications Service Providers (CSPs) are facing challenges represented by the constant need to innovate their IT systems. Meanwhile, in their existing solutions they inherit an integration infrastructure that was often implemented a few decades ago. Using modern software tools, many CSPs undertake modernization initiatives to address flexibility and backward compatibility issues of legacy system integrations.

Some typical roadblocks in any modernization program include:

- How to ensure both newly required and current products are fulfilled correctly in the new solution
    - Systematically map and define existing provisioning flows
    - Define straightforward rules on how to implement new products and keep the future extension costs under control
- How to address integration to existing legacy systems
    - Find a way to ensure *obsolete protocols are integrated*
    - *Reduce integration OpEx costs* by eliminating intermediate integration platforms like InstantLink, homegrown, or 3rd party custom-built solutions
- How to define and execute a cost efficient test strategy
    - The *testing complexity* of service activation is the product of several factors like activation order content, product catalog content, the number of external systems, and existing customer service portfolio. Traditional testing approaches lead to an excessive number of test cases and related efforts.

While TIBCO Fulfilment Orchestration Suite (FOS) and TIBCO integration tools are well suited to implement a modern and dynamic catalog-driven *Service Order Management* system, and Behaim's Resource Order Management Accelerator for FOS complements it in the areas of project methodology, overall system setup, corner integration cases, and smart testing strategy for *Resource Order Management* for activation and management of network elements.

Behaim's ROM Accelerator for FOS is based on numerous successful TIBCO FOS implementations performed by Behaim IT's consulting team. This experience covers the entire project lifecycle starting with the ideation phase and ending with operational support.

The actual project implementation will be precisely defined. Efforts are spent on customer specific situations and common project resources are already in place.

The Behaim ROM Accelerator is comprised of:

- Behaim's Best Practice Methodology
- Ready to Use Code Templates
- Pre-Prepared Document Layouts
- Value Added Tools and Components

The ready-to-use source code is provided as projectware, so it is open to modifications and ready to be further maintained by the customer, for example, in DevOps mode.

2.1. Terms and abbreviations

TERM	DESCRIPTION
TIBCO FOS	TIBCO Fulfillment Orchestration Suite consists of: • TIBCO Product and Service Catalog that can be used to model commercial products, technical services, and their associated fulfillment processes • TIBCO Offer and Price Engine, a commercial offer and pricing server • TIBCO Order Management, a catalog-driven order management system • TIBCO Product and Service Inventory, a system of record for subscriber's inventory
ACCELERATOR	Behaim's ROM Accelerator for FOS
PRODUCT	TM Forum's term for the business entity that is sold to the communications service provider's customer. In TIBCO FOS, a commercial product is called an offer.
SERVICE	In accordance with TMF, a product is analyzed into services, customer-facing or resource-facing

TERM	DESCRIPTION
RESOURCE	In accordance with TMF, this is synonymous with the network element that needs to be activated in order for the customer's communications service to work.
BACKEND SYSTEM	System providing service(s) to customer
NETWORK ELEMENT	Synonym of backend system used by telecommunication service providers. A network element is a resource, per TMF.
SERVICE INVENTORY	Data storage keeping the image of existing customer services and arameters.
PROVISIONING	Calling backend system API to enable or disable services to customer.
PROCESS COMPONENT	Customer specific code acting as a basic building block of plan execution.
INTEGRATION FACADE	Integration process converting one backend system specific API to common API called by process component.
NETWORK ELEMENT	Provisioning API of backend system integrated to FOS based solution.
API	Application programming interface.
TELNET	Communication protocol.
SSL	Communication protocol.
CORBA	Communication protocol.
MML	Man machine language. General term used to indicate text based communications initially intended to be operated by humans.

Table 1 Terms and abbreviations

3. Accelerator description

The accelerator is comprised of:

- Analysis, system design, implementation and testing approach methodology
- Ready-to-use document templates
- Setup of core FOS components
- Structured framework to build the process components
- Reusable integration to common network element types
- Supplementary tools

The ready-to-use source code is provided as projectware, so it is open to modifications and ready to be further maintained by the customer, for example in DevOps mode.

3.1. Project methodology

As-Is Analysis

The order fulfilment project starts with an analytical phase where the as-is state of the system is described.

- The very first step is to align the terminology, e.g., the meaning of the terms products, subscriptions, CPES, customer facing services, value added services, etc.
- The final number of connected network elements in scope is identified. All scenarios (activate, deactivate, suspend, resume, port in/out, MSISDN swap…) to be supported by particular orderable items are identified.
- The cross reference between top level products and backend system involvement in scenarios is documented.
- The available documentation to the existing integration of network elements is identified.
- Any potential need for collaboration with third-party vendors is identified.

High-level design

The aim of the high-level design is to establish the structure of solution. Recommended referential architecture is used as a base for system design. The following items are designed:

- Product and service inventory, either by identifying pre-existing systems or designing a completely new service inventory system
- TIBCO Product and Service Catalog features used to model particular provisioning flow types
- Functionality that must be implemented within process components' code
- Standardized data retrieval for process components data
- Integration mechanisms and protocols that will be used for particular network elements. The APIs of inbound systems, the summary of each network element's operations and features are documented.

Detailed design

The actual catalog content is created. The execution plan fragments and their responsibilities are defined. Integration mapping to the frontend and backend systems are documented. Operations tools like logging and error handling mechanisms for automatic and manual recovery are designed. Guidelines of messaging system use are set.

The eventual build process is not affected by the methodology. The implementation team is free to use its own approach, whether it is waterfall-style project management or agile-focused work organization.

Testing approach contribution

The systematic approach to the solution, the use of FOS capabilities, and the consolidation of process components shall result in a significant reduction of user acceptance testing efforts that will cover the complete system functionality. The accelerator bets on reusable automated testing.

The accelerator methodology and tools support all testing phases

- Unit tests. Unit testing, including the automation of regression testing, is continuously performed by the CI/CD framework. Unit test data is already captured in the mapping documents.
- Integration tests. The goal is to enable an isolated integration test without code change.
- Acceptance test. Addressing the permutation like complexity and potential large number of test cases in acceptance test

Integration adapters, whether towards the backend systems or the order intake systems allow the isolation of an operation from the rest of the solution. The unified data logging supports one-at-a-time integration testing.

The FOS implementation team closely cooperates with the test manager and the test analyst to structure the acceptance tests in order to balance a reasonable number of test cases to be performed and the solution coverage ratio.

The optimal result is achieved by leveraging systematic resource model design through the use of the TIBCO Product and Service Catalog, by collecting input data for process components in a unified method, and by approaching integration uniformly using integration adapters.

3.2. Ready-to-use document templates

Behaim's ROM Accelerator for FOS comes with a number of document templates to support the progressive phases of the entire project lifecycle:

- Solution blueprint document template
- High level design document
    - Responsibility split between integration and process logic
    - Introduction of the resource adapter concept
- Process Component Technical Design Template
- Integration Facade Technical Design Template
    - Data mapping document templates

- Definition of actual catalog content
    - Definition of products, features used
    - Definition of execution plan fragments and their responsibilities
- Design of operations tools
    - Logging and error handling mechanisms for automatic and manual recovery
    - Guidelines of messaging system use
    - Performance considerations of integration code

3.3. Setup of core FOS components

Assumed TIBCO product stack

The complete solution assumes the deployment of the following TIBCO products. The product stack can be divided into two groups based on their roles in any FOS implementation project.

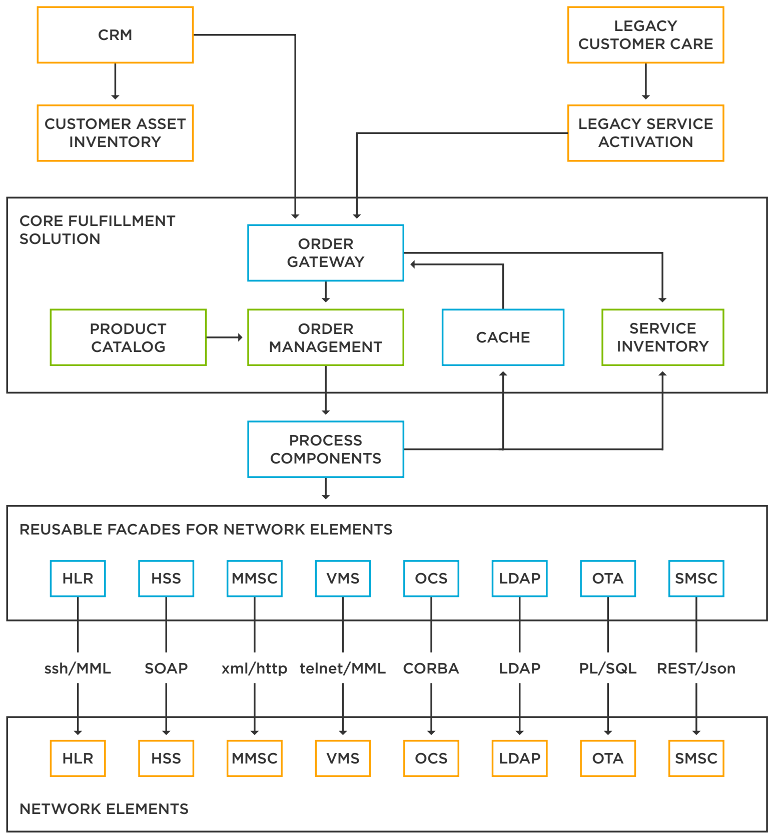
Order fulfillment software

- TIBCO Product and Service Catalog
- TIBCO Order Management, for order decomposition and orchestration
- TIBCO Product and Service Inventory to act as a repository for service inventory

Integration software

- TIBCO EMS messaging system connecting the individual subsystems
- TIBCO BusinessWorks software integration tool
- TIBCO ActiveMatrix BPM business process management tool for fulfillment tasks involving human interaction
- TIBCO ActiveSpaces distributed in-memory cache

All components are installed in a fail-over configuration.



While TIBCO FOS is a proven Service Order Management system, Behaim IT has developed resource-facing integration facade templates, along with implementation artifacts that enable FOS to be used concurrently as a catalog-driven Resource Order Management system. Using this strategy, the customer can reduce upfront capital expenditure, simplify deployment footprint, and reduce ongoing maintenance costs.

3.4. Structured framework to build the process components

Process components are provided in the form of code skeletons for typical use cases. Components are structured to the parts.

- Reading data, typically parameters and characteristics from own order line, different order lines, work context, service inventory, and orchestration specific information like affinity-based IDs
- Executing command(s) using backend system facades

An asynchronous execution pattern is provided in the example implementation. In the template implementation, a shared library for common logging and error handling components is used.

3.5. Reusable integration to common network element types

Integration Facade Code templates with common interface to process components will avoid the need to use intermediate integration platforms (like InstantLink) in the final solution.

A wide set of existing templates connecting to the common backend system running their APIs on a broad variety of protocols is available. This includes specific legacy-oriented templates:

- MML protocol over telnet or ssh lines is supported by TIBCO BusinessWorks, using especially developed TTY access plug-In with Groovy scripting capability for network elements like Ericsson HLR, Nokia MSC, VMS servers, and similar text-based legacy systems.
- CORBA based communication is supported by Java-based code called from TIBCO BusinessWorks.
- Oracle DB direct access or PL/SQL access is supported by TIBCO BusinessWorks JDBC activity.

Up-to-date protocols are also handled by the template set:

- SOAP-based communication to systems with more up-to-date provisioning interfaces like Nokia PGW
- Rest/JSON based communication
- Pure XML over HTTP(s)

3.6. Supplementary tools

Supplementary project supporting tools are provided as well. They include:

- Error handling and reprocessing component, including client process shared library, DB scripts GUI, and server side process
- Code Template for order intake facades (north side of system)
- Code Template for interfacing with service Iinventory

3.7. Prepared projectware components

Although modern software developers prefer REST APIs, based on Behaim's experience from many TIBCO Order Management implementations, about 50% of network element interfaces are still on legacy interfaces:

- CORBA
- telnet
- ssh

where MML is emulated over telnet and ssh.

3.7.1. CORBA integration

CORBA interfaces are implemented using Java Global Instance and Java Invoke Method activities.

To reach CORBA server object implementation from TIBCO BusinessWorks, the following steps are required:

- Running *idl2java* compiler to get stub classes for modules; their methods are defined in an IDL file
- Compilation of generated Java files, containing stub classes for accessing CORBA servers
- Instantiating CORBA client objects and storing the instances into Java Global Instance
- Using Java Methods activities to call the methods of the CORBA client object instances stored in Java Global Instance resources

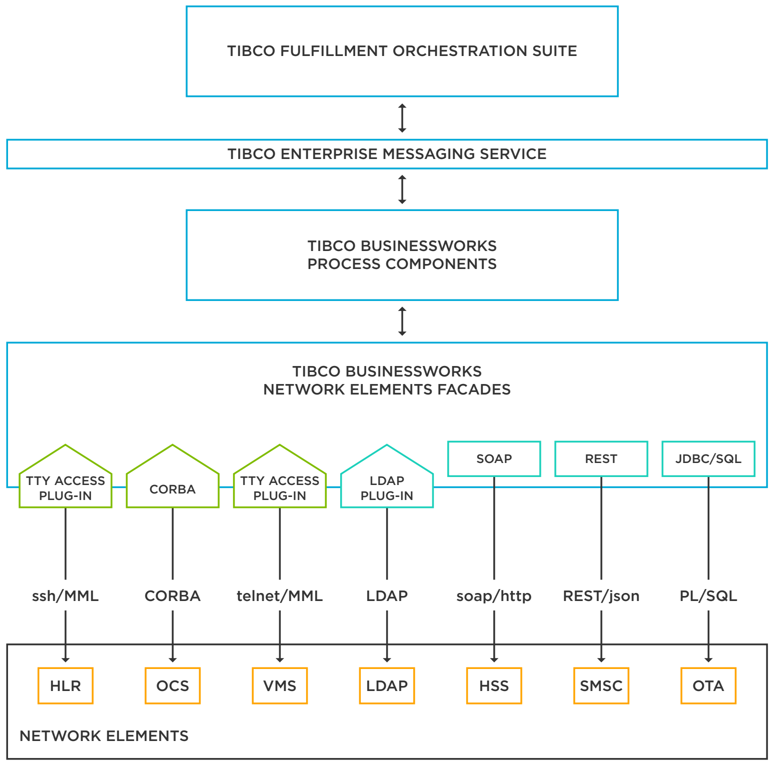
To make the integration of CORBA objects seamless and visible in TIBCO BusinessWorks, the accelerator uses *method2object* utility. The utility generates:

- For each interface method, a serializable method object is generated where the method parameters become fields of the member method
- For each module defined in IDL file, the utility generates a TIBCO BusinessWorks wrapper. The wrapper methods use the generated method objects as parameters. The original methods are called from the methods of the TIBCO BusinessWorks wrapper object.
- Method objects can be then used by *XML* to *Java* and *Java* to *XML* activities where parameter names are visible and provides better view into the interface for a TIBCO BusinessWorks designer/developer.

3.7.2. TTY access plug-in

TTY access plug-in consists of two components:

- *TIBCO BusinessWorks Plug-in for TTY software* enables the integration of Network Element over telnet or ssh emulating terminal operating personnel shortly called MML interface
- *TTY access utility* is a tool that helps to develop, tune, and test the communication with NE elements. It makes it possible to fine tune the Groovy script that controls command-rendering and external system response-parsing in the MML communication, without the need to run the whole setup, including the deployment of integration facade. It significantly increases productivity during integration testing.



4. Developing TIBCO BusinessWorks application using TTY plug-in functionality

Behaim TIBCO plug-in for TTY allows the execution of commands and scripts on a remote server. It supports two basic transport protocols, i.e., Telnet and SSH. In the background of the plugin there is a session manager, which takes care of the sessions' lifecycle. It is possible to create many sessions at once.

The following schema shows TTY plug-in architecture:

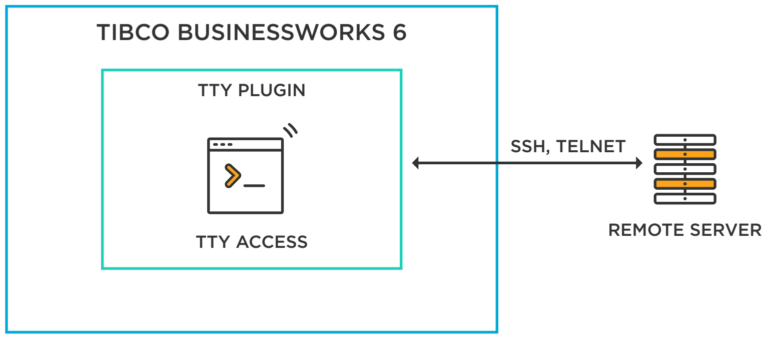


Figure 1 Behaim TIBCO Plug-in TTY Architecture

The TTY plug-in palette contains activities listed below

- Destroy
- EndFlow
- Init
- RemoteCommand
- StartFlow
- TTYAutomation

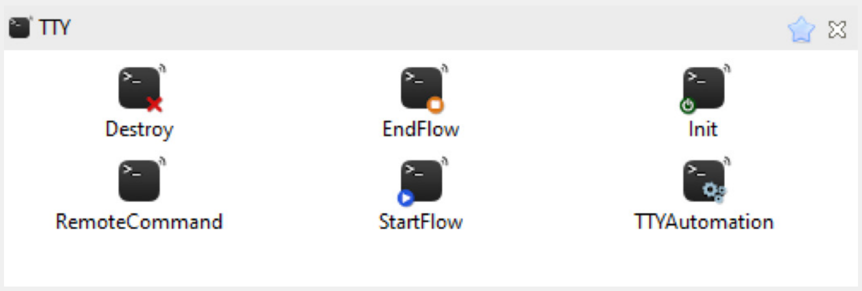


Figure 2 Palette activities

This section describes how to configure the TIBCO BusinessWorks application with TTY plug-in functionality in TIBCO BusinessStudio for BusinessWorks.

All supported features are available within the TTY palette.

4.1. TTY connection management

Connection to the remote server is managed by Init and Destroy activities. The recommended approach is to create a process activator with Init activity for establishing a connection and Destroy activity for deleting a connection.

4.1.1. Remote server connection configuration

Init activity holds connection configuration such as *host, port, protocol, target system, and authentication.*

Host field serves as endpoint address (e.g. 10.0.0.53).

Port field serves as port number (e.g 22 or 23).

Protocol field serves as transport protocol (Telnet or SSH).

Target system field serves as the endpoint representation of a new line character.

Authentication indicates type of authentication and provides three options:

- None
- Username and password
- Certificate

Init activity can be configured in general tab or in activity input. The activity input is taking a preference.

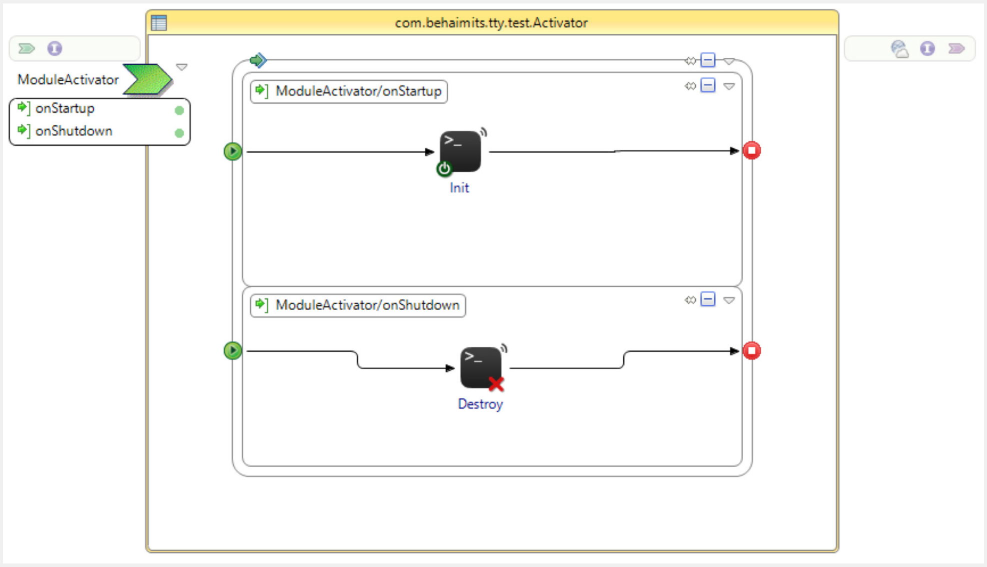

com.behaimits.tty.test.Activator
ModuleActivator
onStartup
onShutdown
ModuleActivator/onStartup
Init
ModuleActivator/onShutdown
Destroy

Figure 3 Activator

4.2. TTY remote command

TTY remote command allows the execute command in a remote server. The result of the command is sent back to TIBCO BusinessWorks activity via stream. Each activity supports executing exactly one command in a remote server.

TTY remote command consists of one activity called *RemoteCommand*.

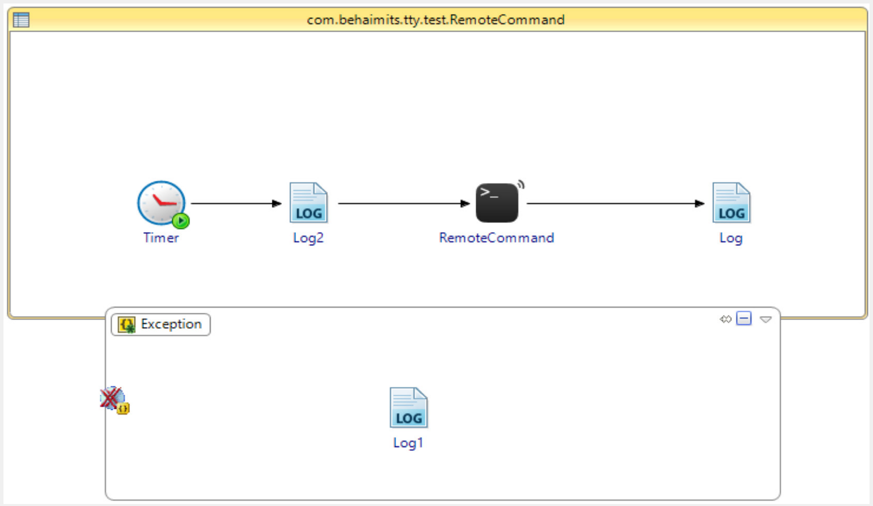


Figure 4 Process with RemoteCommand activity

4.3. TTY execute script

TTY execute script allows the execution of the Groovy script. Every Groovy script accepts parameters such as:

- Expected – list of expected values
- Return – return object serialized into XML format
- Identifiers – List of pair (key–value)
- Arguments – List of pair (key–value)

Communication with the server is enabled via TTYaccess class object, which allows access to the server from the *Groovy* script. TTY access also implements the method *expect*. The method accepts a list of *regex* patterns and reads lines from the input stream until one of the patterns matches. The obtained lines can then be parsed in the Groovy code as Groovy has very strong support for regular expressions and parsing strings.

The result is obtained in key–value form. Every expected item from input will be passed to output with or without its value.

TTY execute script consists of one activity called *TTY automation*.

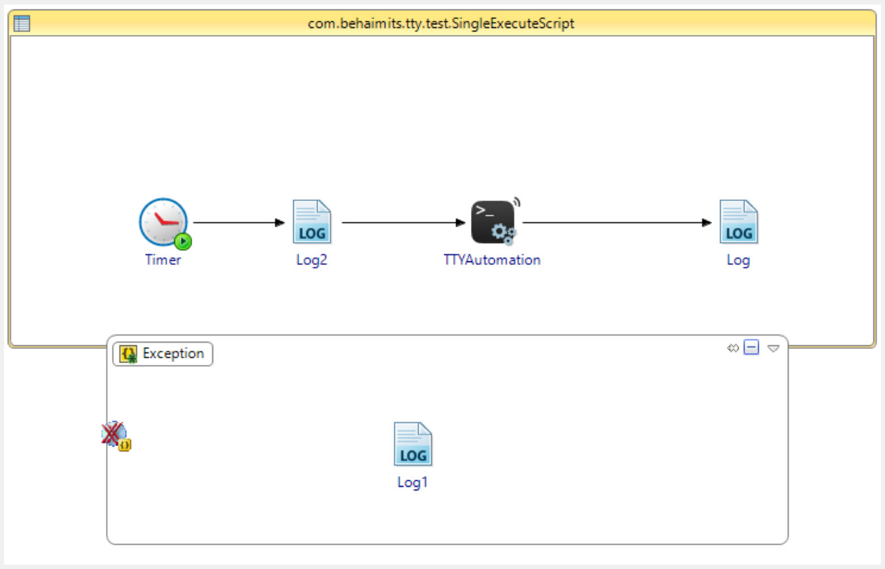


Figure 5 Process with TTY automation activity

The sample below demonstrates the script's features.

```
import java.util.regex.Matcher

tty.setNl("\r\n")
class Parser {
    private RESP = [:]
    private expectMap = [
        PERMANENT_SUBSCRIBER_DATA :'PERMANENT
SUBSCRIBER DATA.*$',
        SUBSCRIBER_IDENTITY:'SUBSCRIBER IDENTITY.*$',
        SUPLEMENTARY_SERVICE_DATA:'SUPPLEMENTARY SERVICE
DATA.*'
    ]

    void parse() {
        def sec = tty.expectRegex(expectMap)

        def i=0
        while(sec != null) {
            i++
            if (sec == 'PERMANENT_SUBSCRIBER_DATA' ) {
                parsePermSubscData()
            } else if (sec == 'SUBSCRIBER_IDENTITY') {
                parseSubscriberIdentity()
            } else if (sec == 'SUPLEMENTARY_SERVICE_DATA') {
                parseSuplementatryServiceData()
            }
            sec = tty.expectRegex(expectMap)
        }
    }
```

4.4. TTY flow

When using *RemoteCommand* activity or *TTY automation* activity, the session manager selects a classic session. In some cases, it is necessary to execute commands one by one in the same environment. Executing one command depends on the result of the previous command.

TTY provides two activities for creating flow of *RemoteCommand* and *TTY automation* sharing the same session.

TTY execute script consists of two activities:

- *StartFlow* – allocate session for flow communication.
- *EndFlow* – release session back to available sessions.

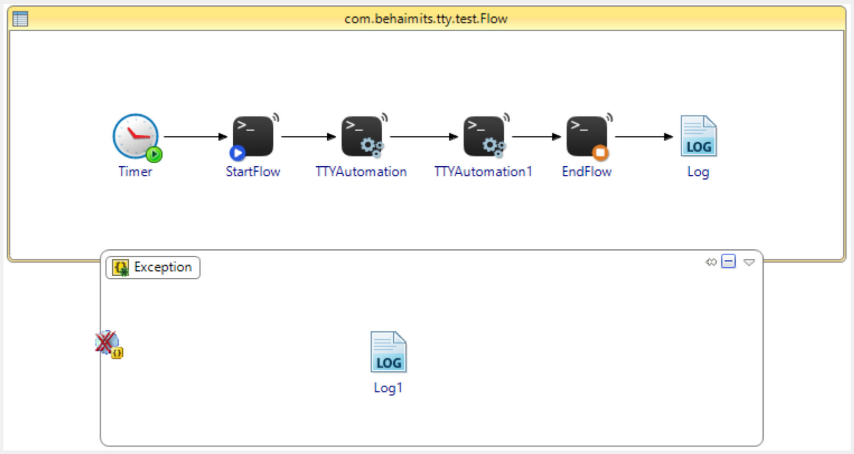


Figure 6 Process with flow sessions

Global Headquarters
3307 Hillview Avenue
Palo Alto, CA 94304
+1 650-846-1000 TEL
+1 800-420-8450
+1 650-846-1005 FAX
www.tibco.com