Thread Adapter

In models with high throughput, it can be helpful to split a task into multiple independent actions. To do that, use Persistent States to create Persistent State objects or model a state machine with fork and join. Modeling Persistent States is easier than managing individual threads. Also, support for tracing, debugging and managing is offered for Persistent State.

With the deprecated **Thread Adapter** you can create threads from a service call (main thread) or independent threads. Each new thread runs an operation parallel to the main thread. Threads do not share any data. Input data to the operation is cloned before a new thread starts. The session handling (commit and rollback) is separated. A thread has no output parameters besides the automatically generated **threadID** (see Thread Adapter Reference). Threads can be created with any amount of input parameters but the called operation will not provide any output parameters.

In the main thread, you can wait for other threads to complete by using the <<WaitForThread>> action. If you don't wait, the other threads will run in the background.

With the **Thread Adapter**, you can

- Create a Thread (deprecated)
- Block a Thread
- Wait for a Thread (deprecated)

Related Pages:

- Thread Adapter Reference
- Persistent States and Signals