Enacting BPEL4WS Specified Workflows with Multiagent Systems

Paul Buhler[1] José M Vidal[2]

[1]Department of Computer Science
College of Charleston

[2]Department of Computer Science and Engineering
University of South Carolina

Workshop on Web Services and Agent-Based Engineering
19 July 2004

UNIVERSITY OF SOUTH CAROLINA

BPEL4WS and Workflows

- The Business Process Execution Language for Web Services is a workflow language.
- XML language.
- Uses WSDL.
- Assumes SOAP.
- Widely adopted.
- Centrally and statically enacted.

GOAL

Enact BPEL4WS workflows using dynamically created multiagent systems.

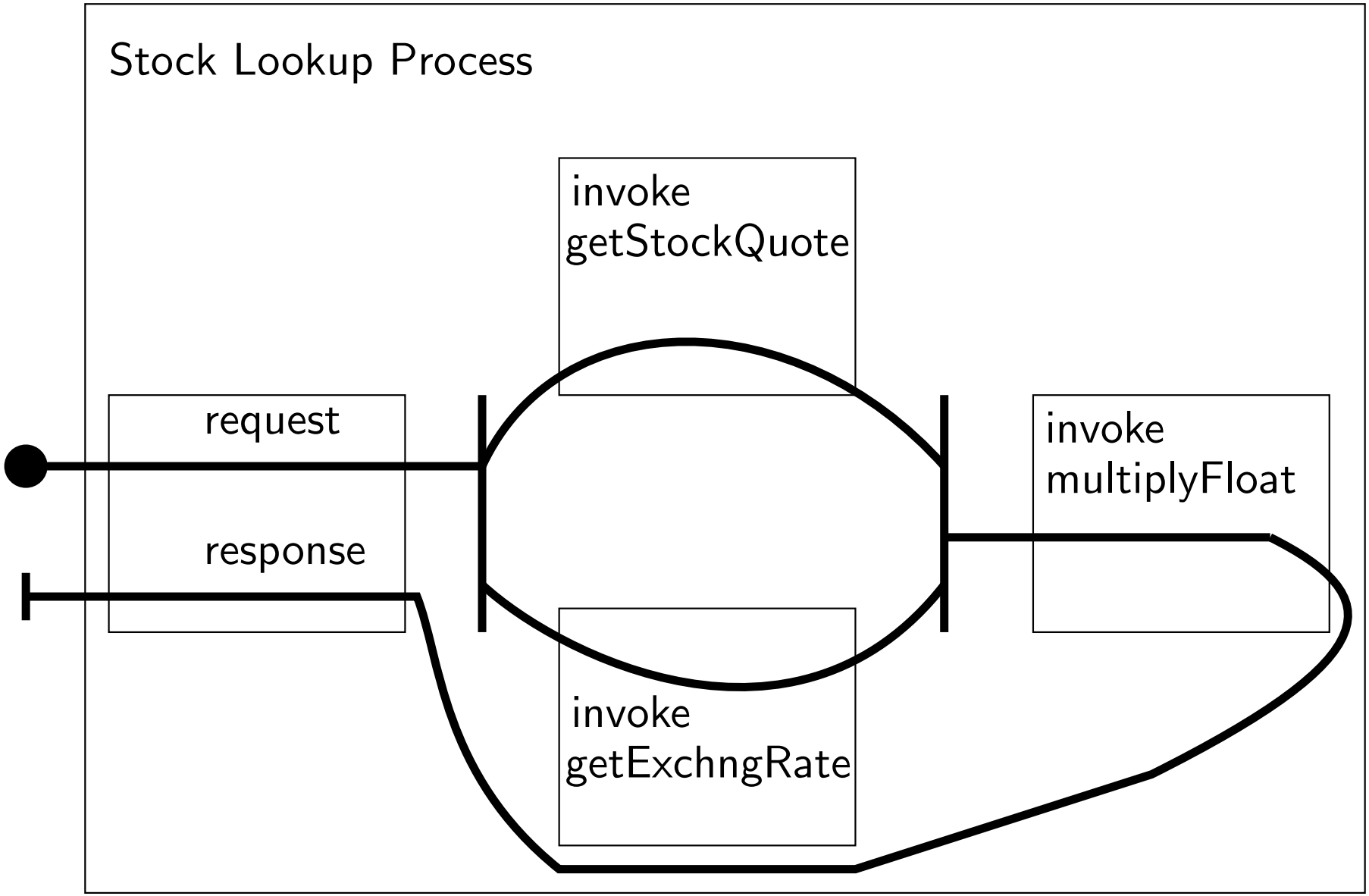
BPEL4WS and Workflows

- The Business Process Execution Language for Web Services is a workflow language.
- XML language.
- Uses WSDL.
- Assumes SOAP.
- Widely adopted.
- Centrally and statically enacted.

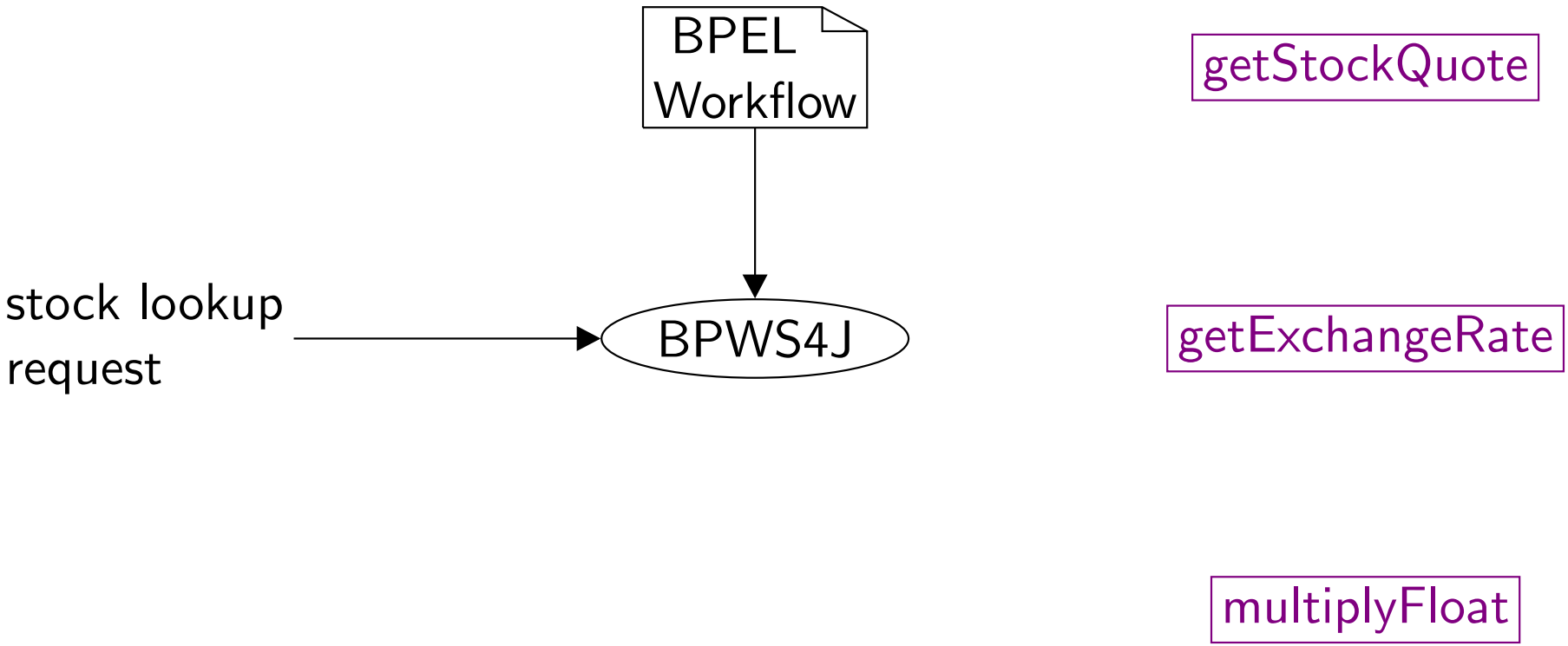
GOAL

Enact BPEL4WS workflows using dynamically created multiagent systems.

Sample Workflow



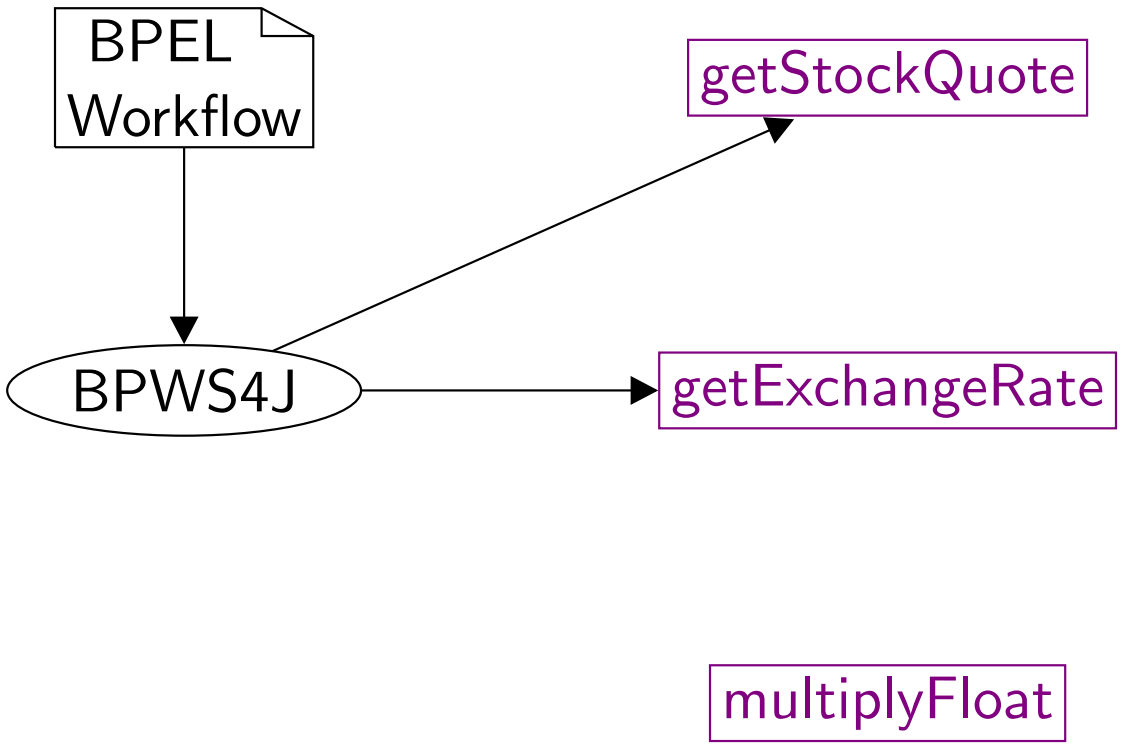
Centralized Enactment



- BPWS4J enactment engine from IBM AlphaWorks.

Centralized Enactment

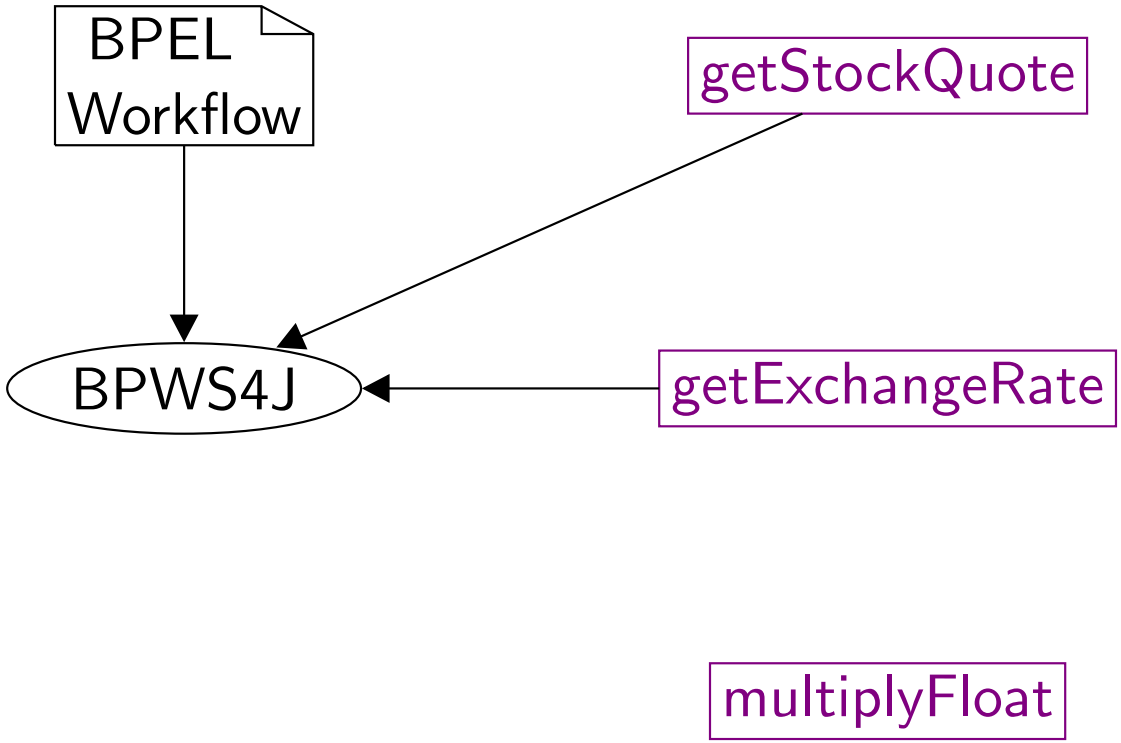
stock lookup request



- BPWS4J enactment engine from IBM AlphaWorks.

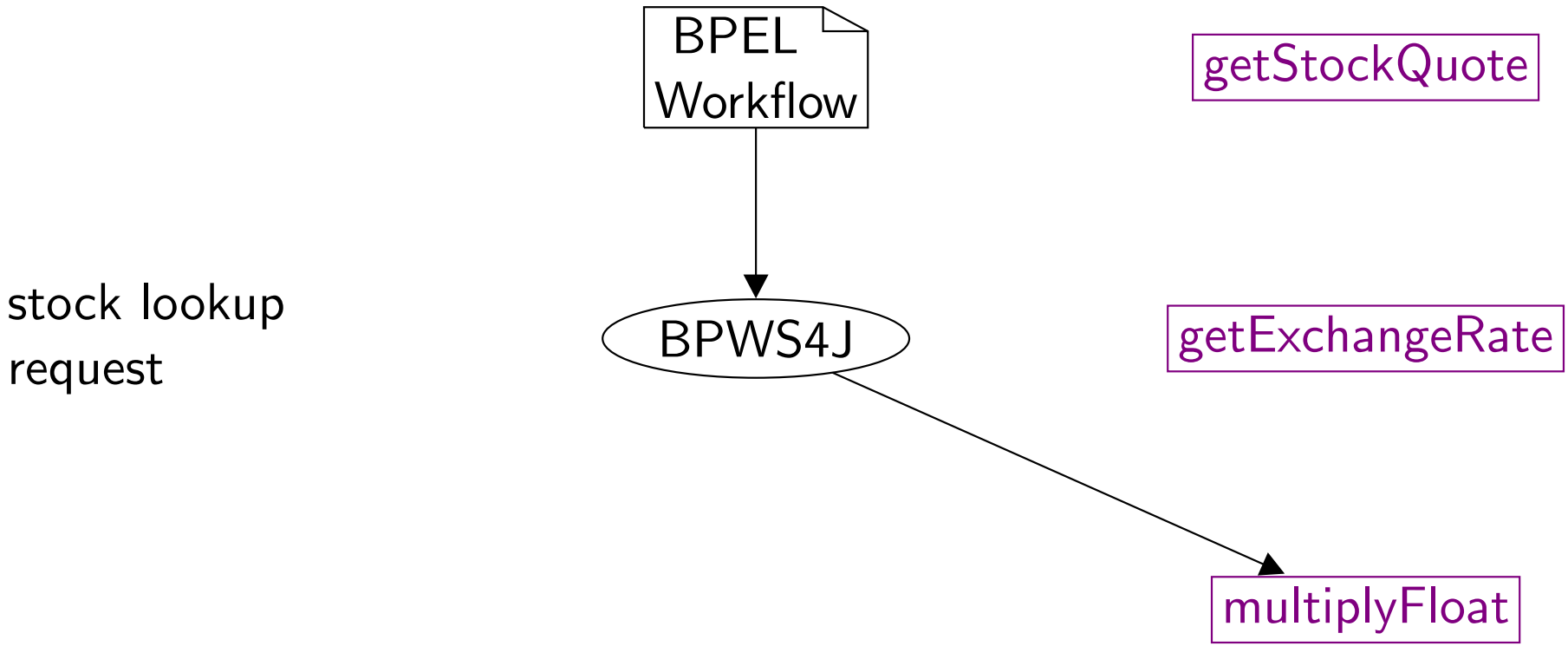
Centralized Enactment

stock lookup request



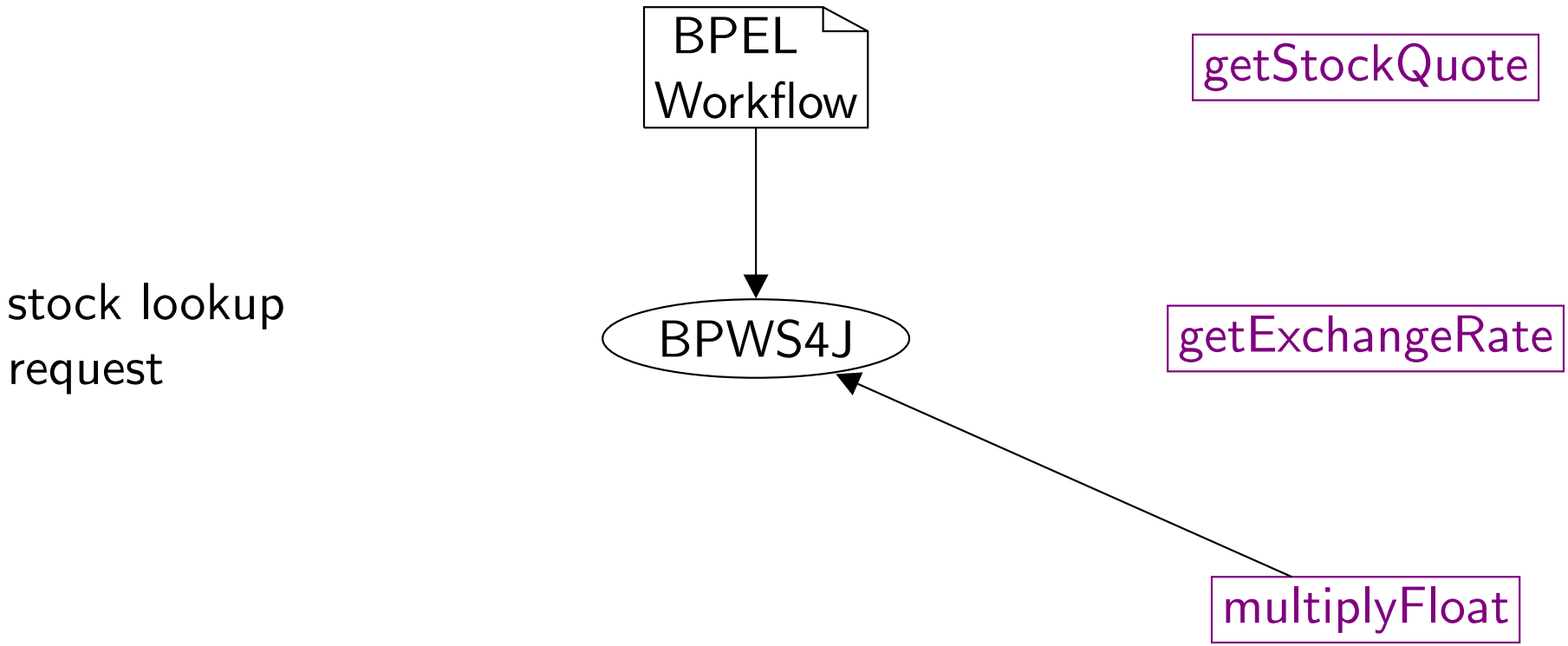
- BPWS4J enactment engine from IBM AlphaWorks.

Centralized Enactment



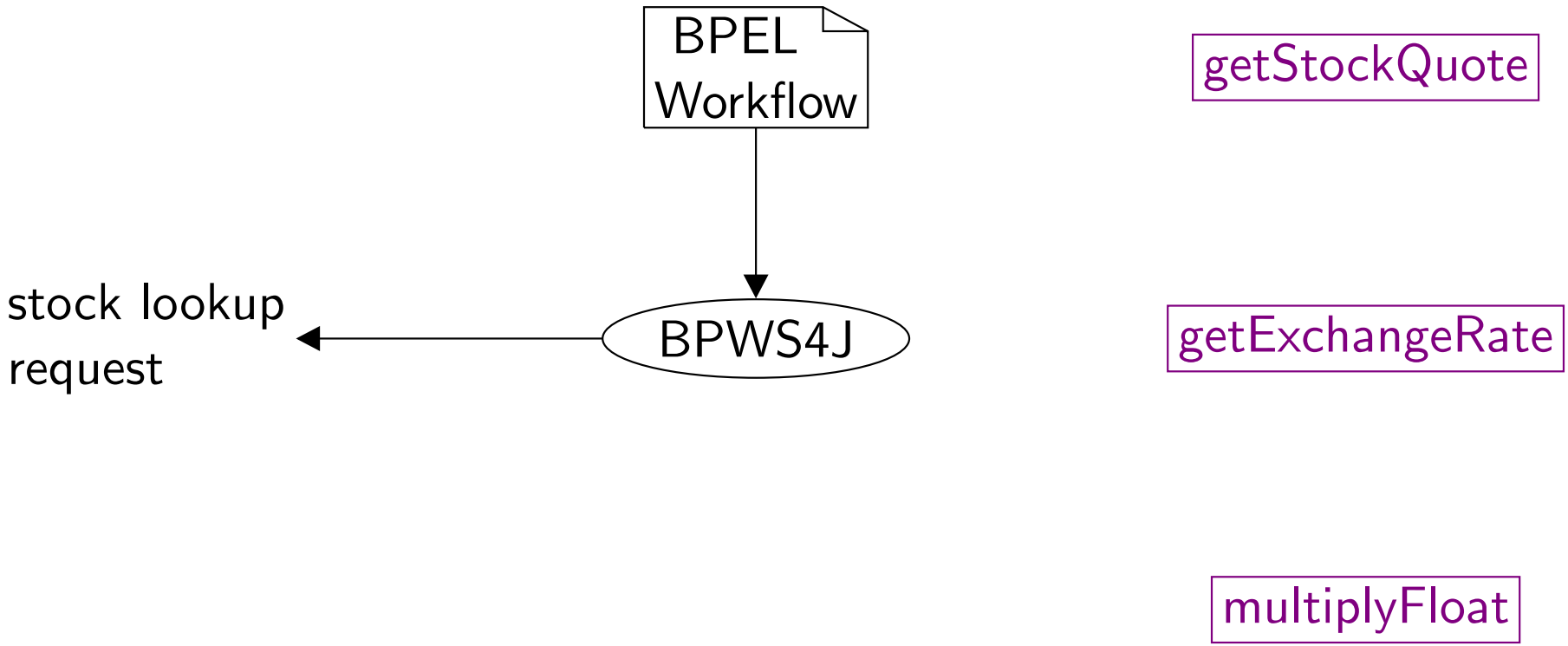
- BPWS4J enactment engine from IBM AlphaWorks.

Centralized Enactment



- BPWS4J enactment engine from IBM AlphaWorks.

Centralized Enactment

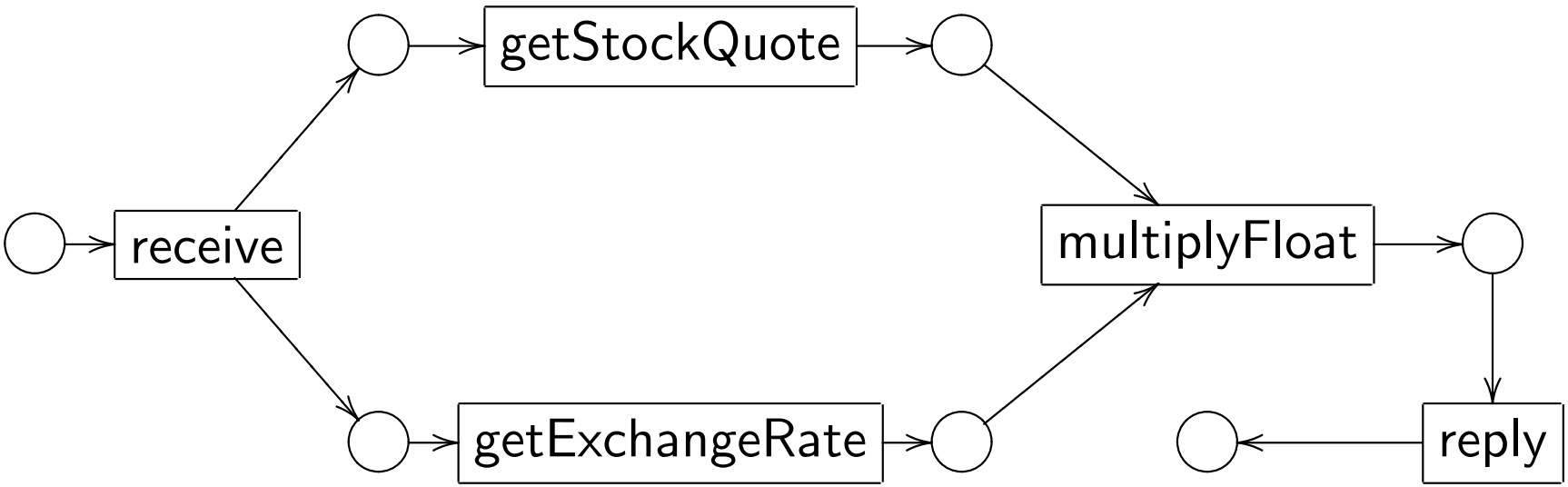


- BPWS4J enactment engine from IBM AlphaWorks.

Centralized to Decentralized Enactment

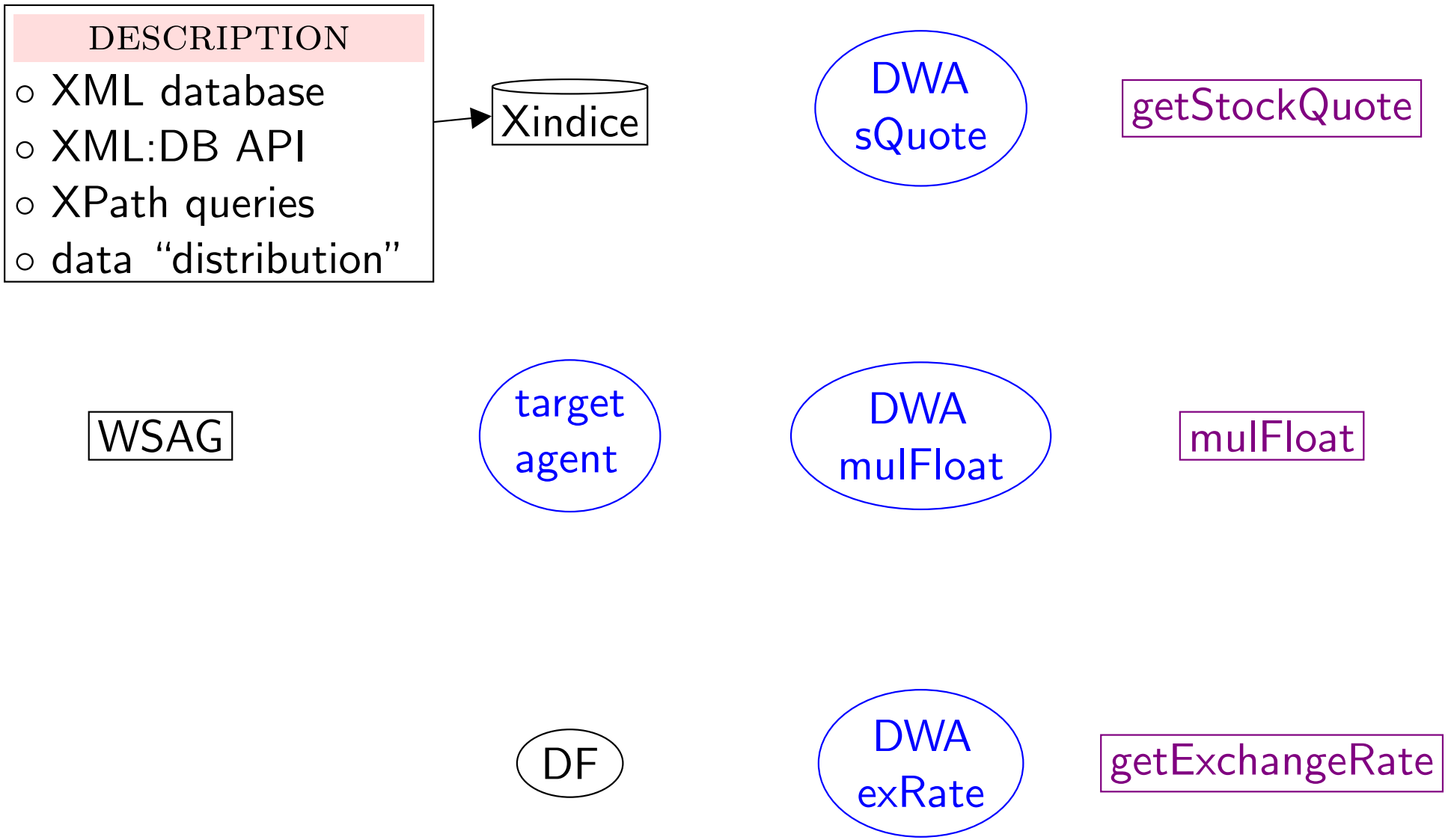
- Needed to coordinate agents.
- Used Colored Petri Nets and a native XML database.
- Agents decide who to call next: distributed control.

Colored Petri Mapping

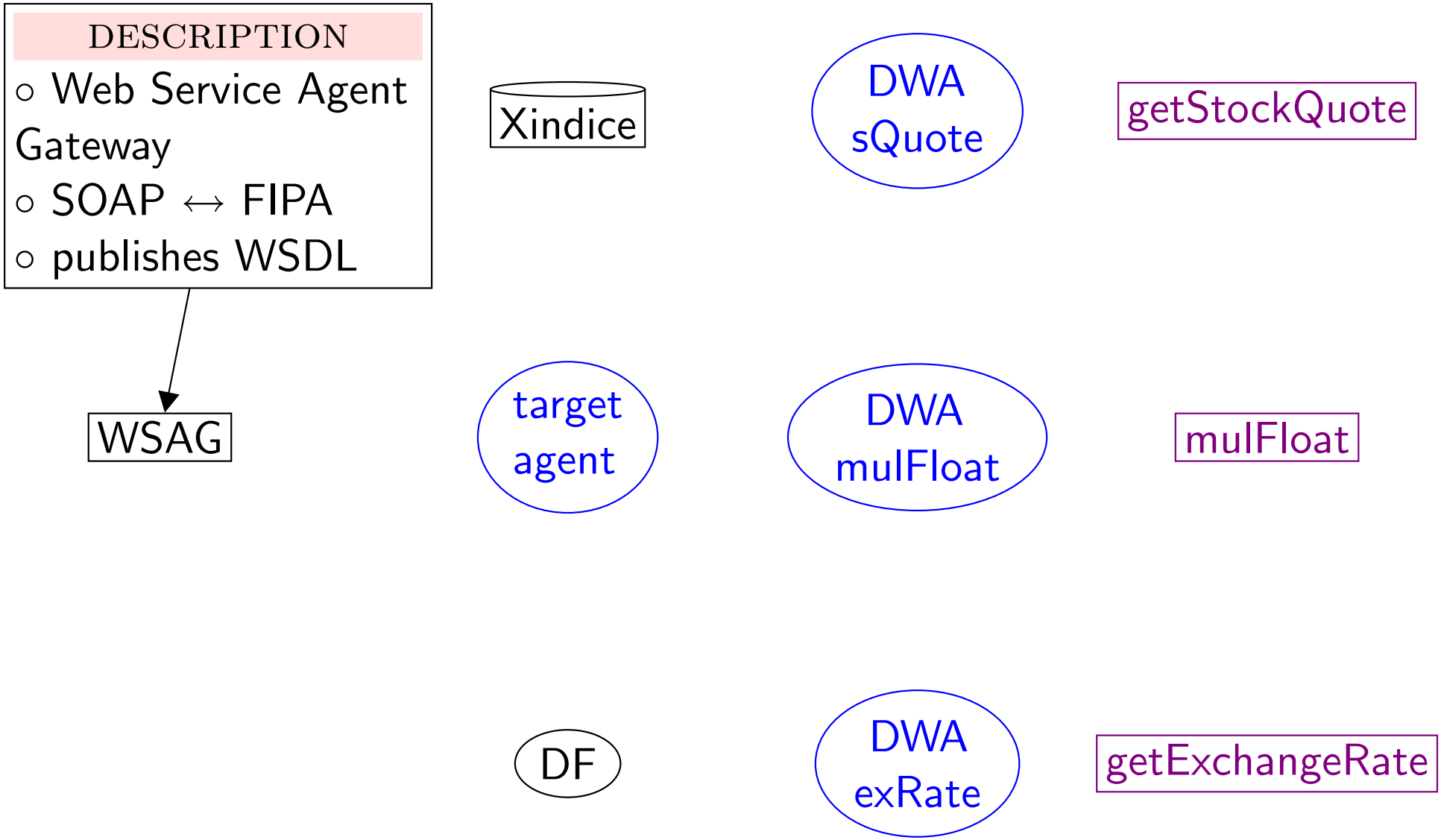


- Tokens carry data: control information.

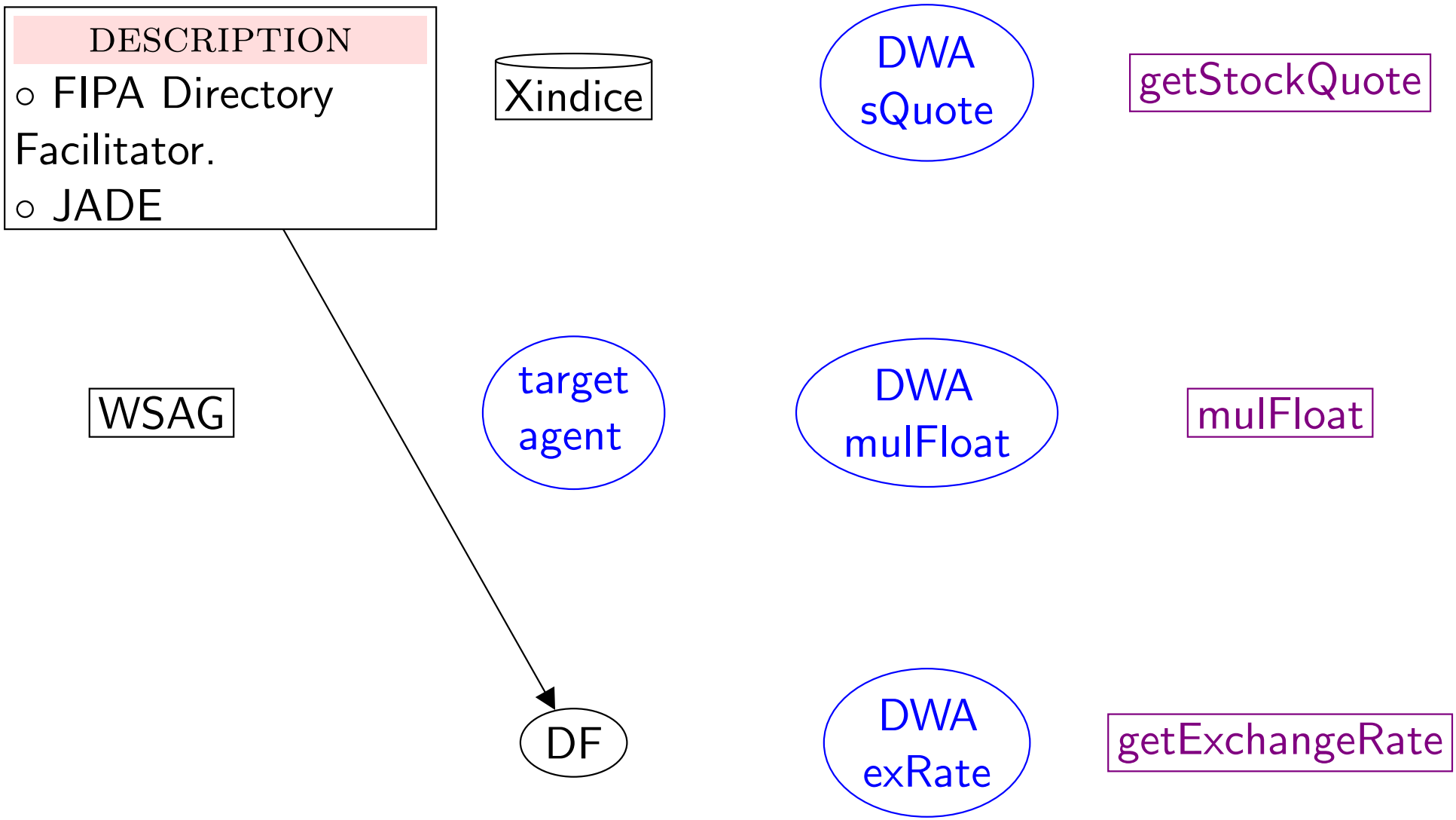
Multiagent Workflow Enactment Example



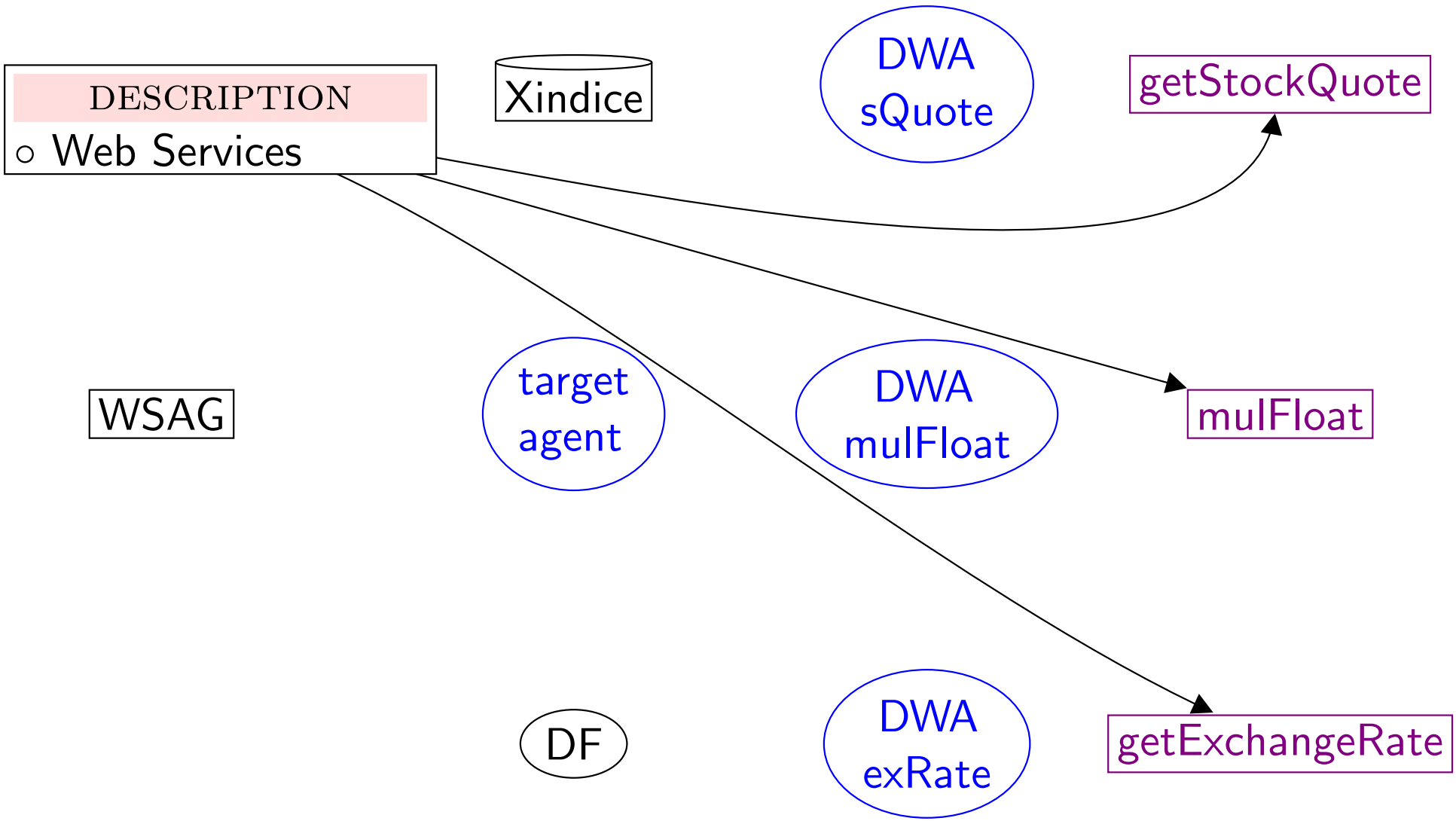
Multiagent Workflow Enactment Example



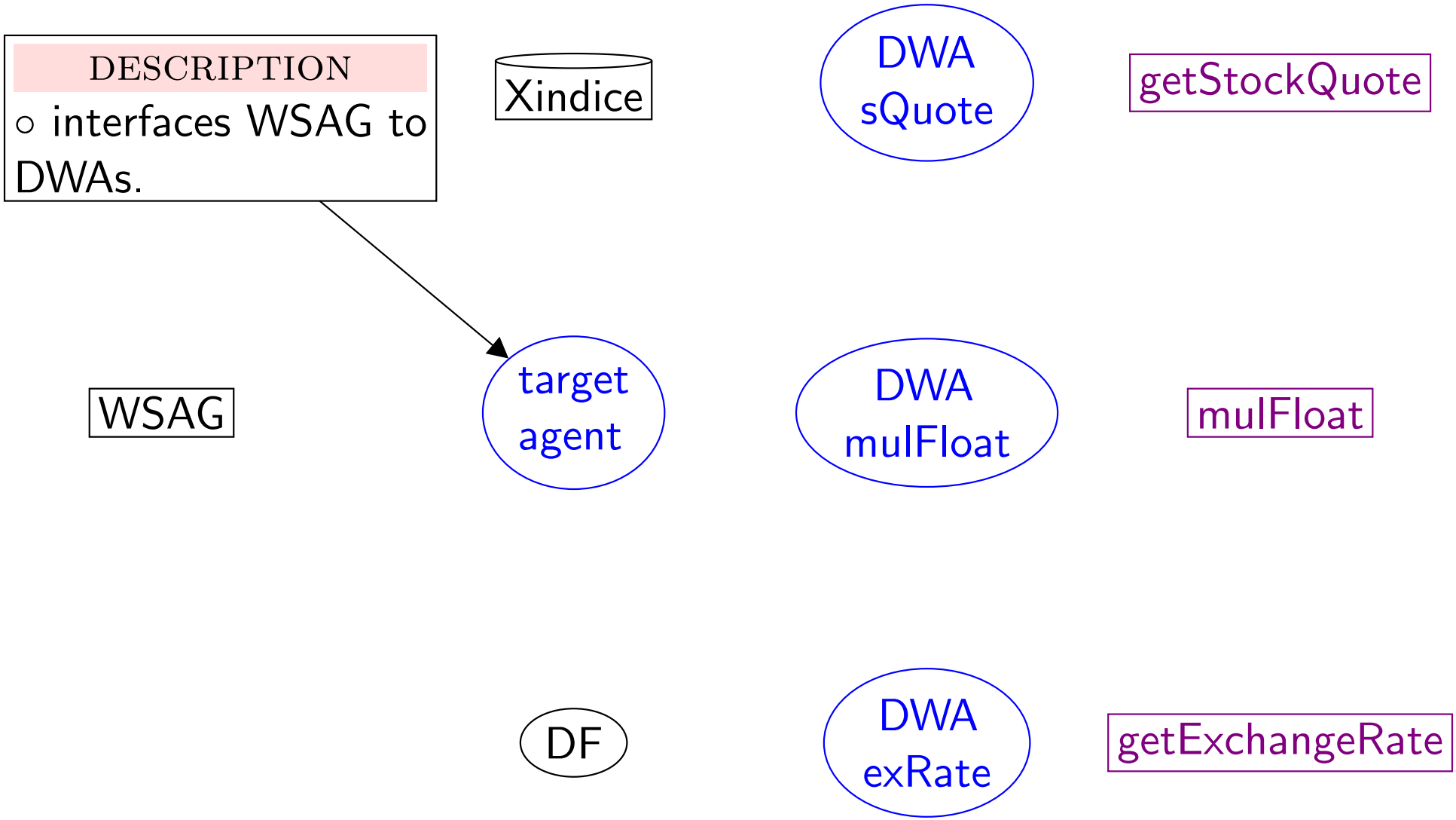
Multiagent Workflow Enactment Example



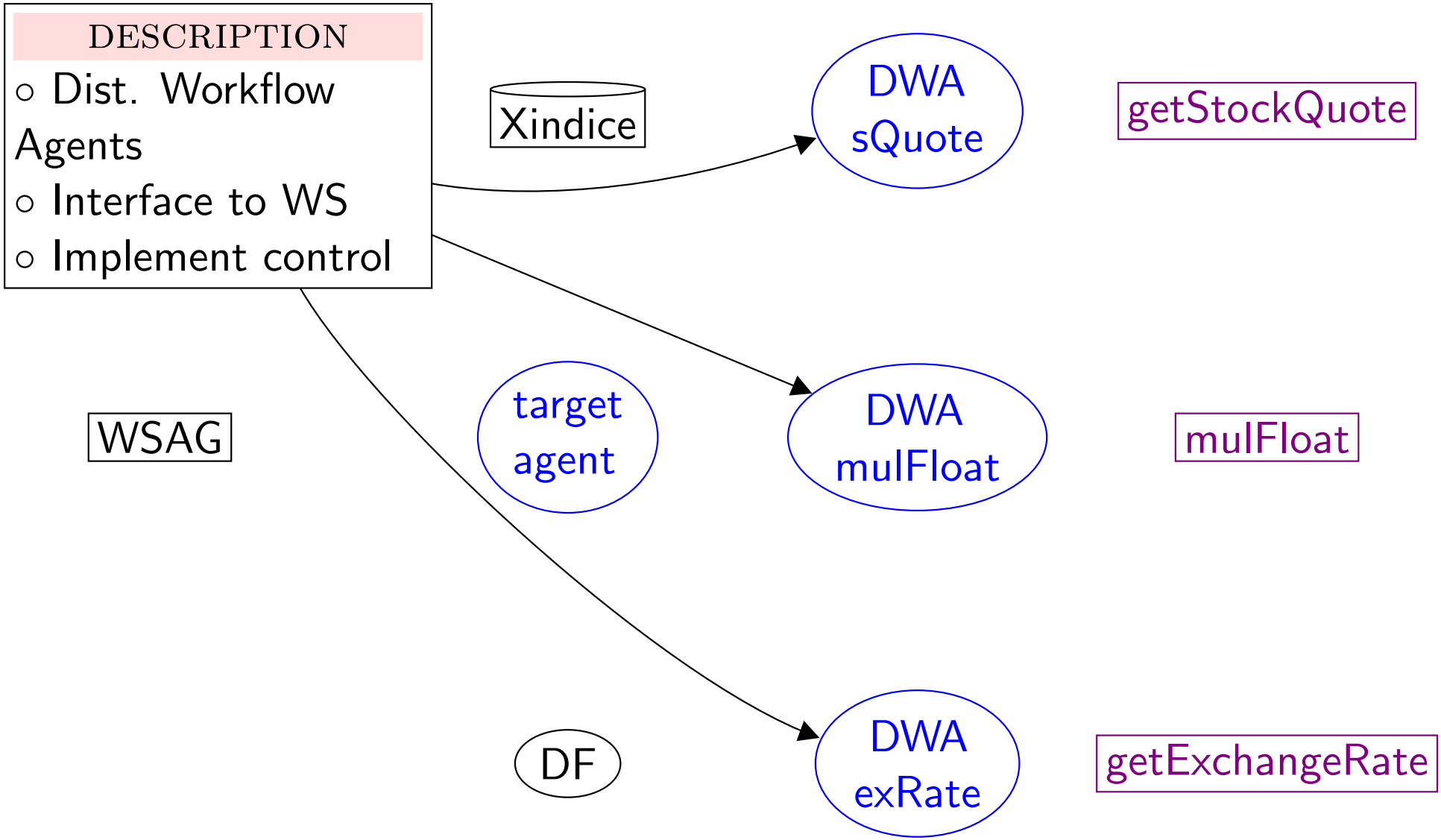
Multiagent Workflow Enactment Example



Multiagent Workflow Enactment Example



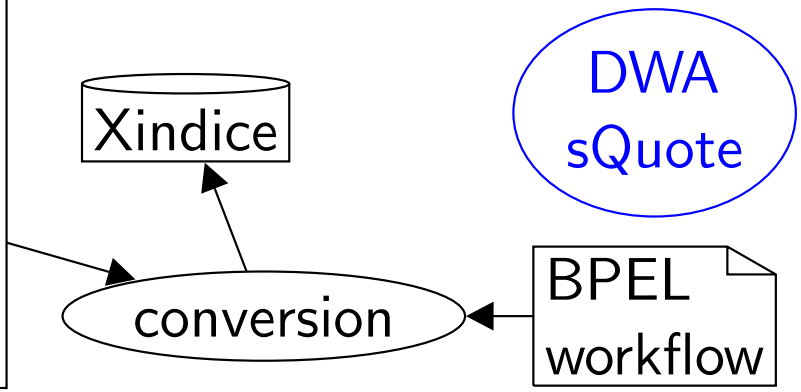
Multiagent Workflow Enactment Example



Multiagent Workflow Enactment Example

DESCRIPTION

- Distributed Wkflow Agent configuration
- could be automated
- need only one agent program



DWA
sQuote

getStockQuote

WSAG

target
agent

DWA
mulFloat

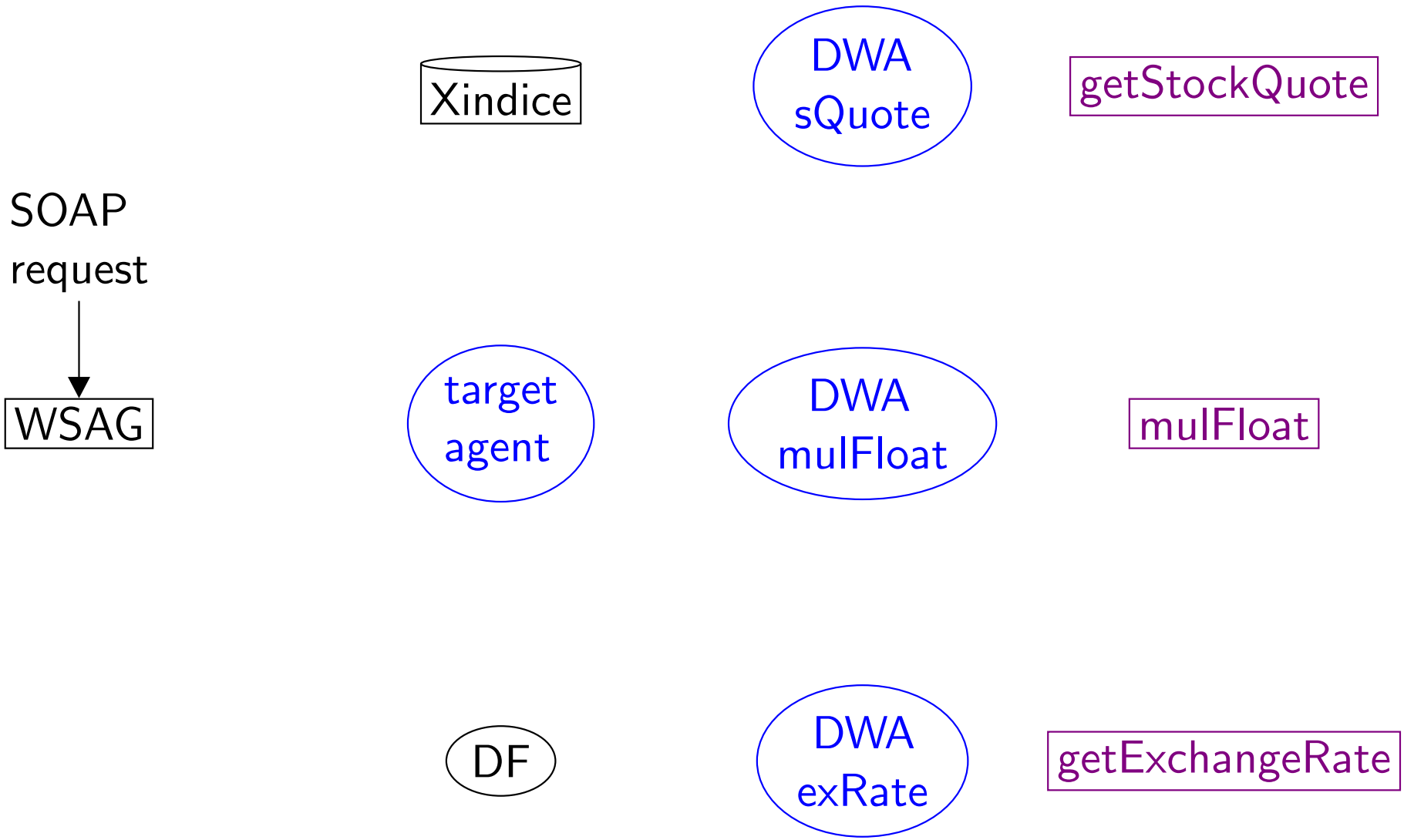
mulFloat

DF

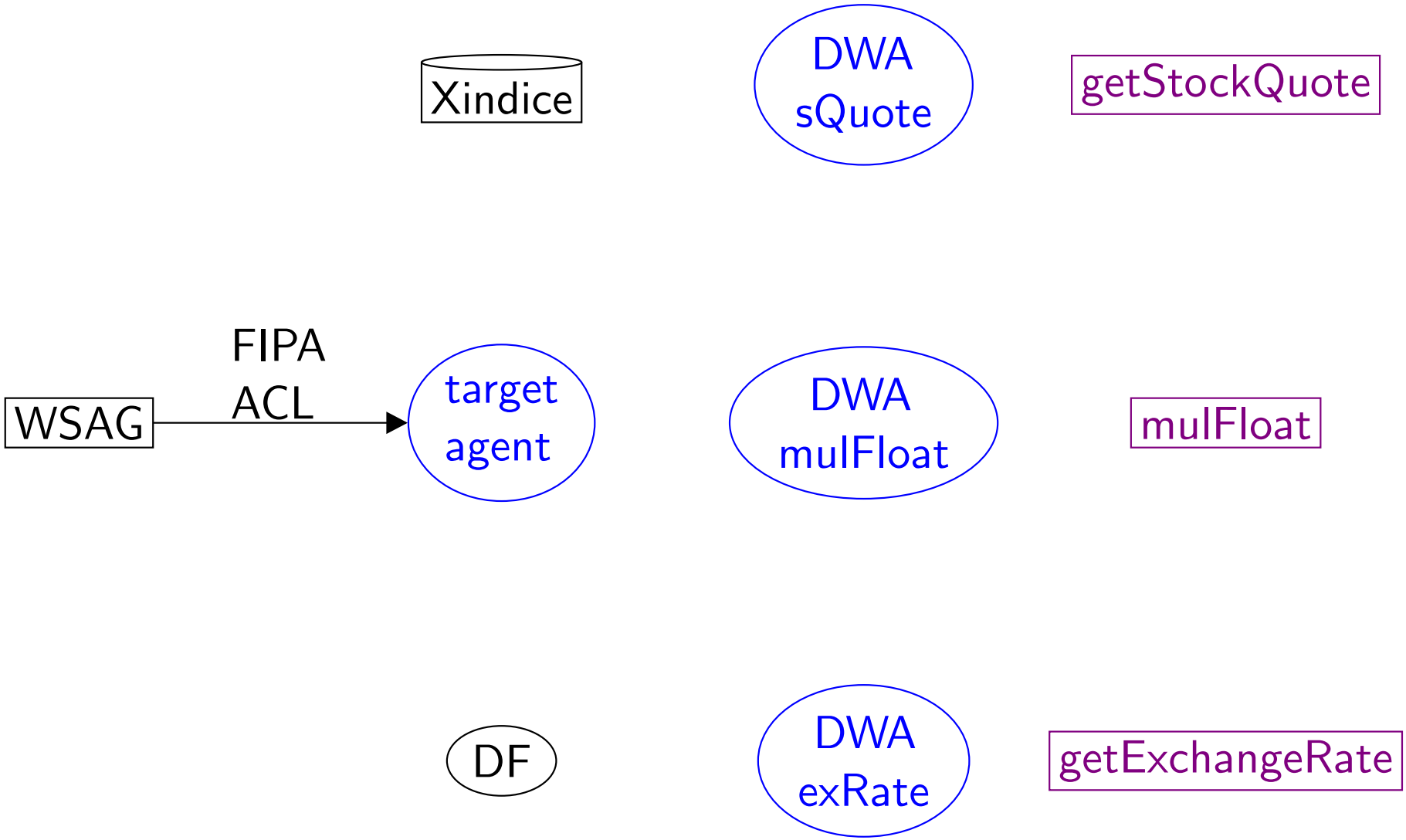
DWA
exRate

getExchangeRate

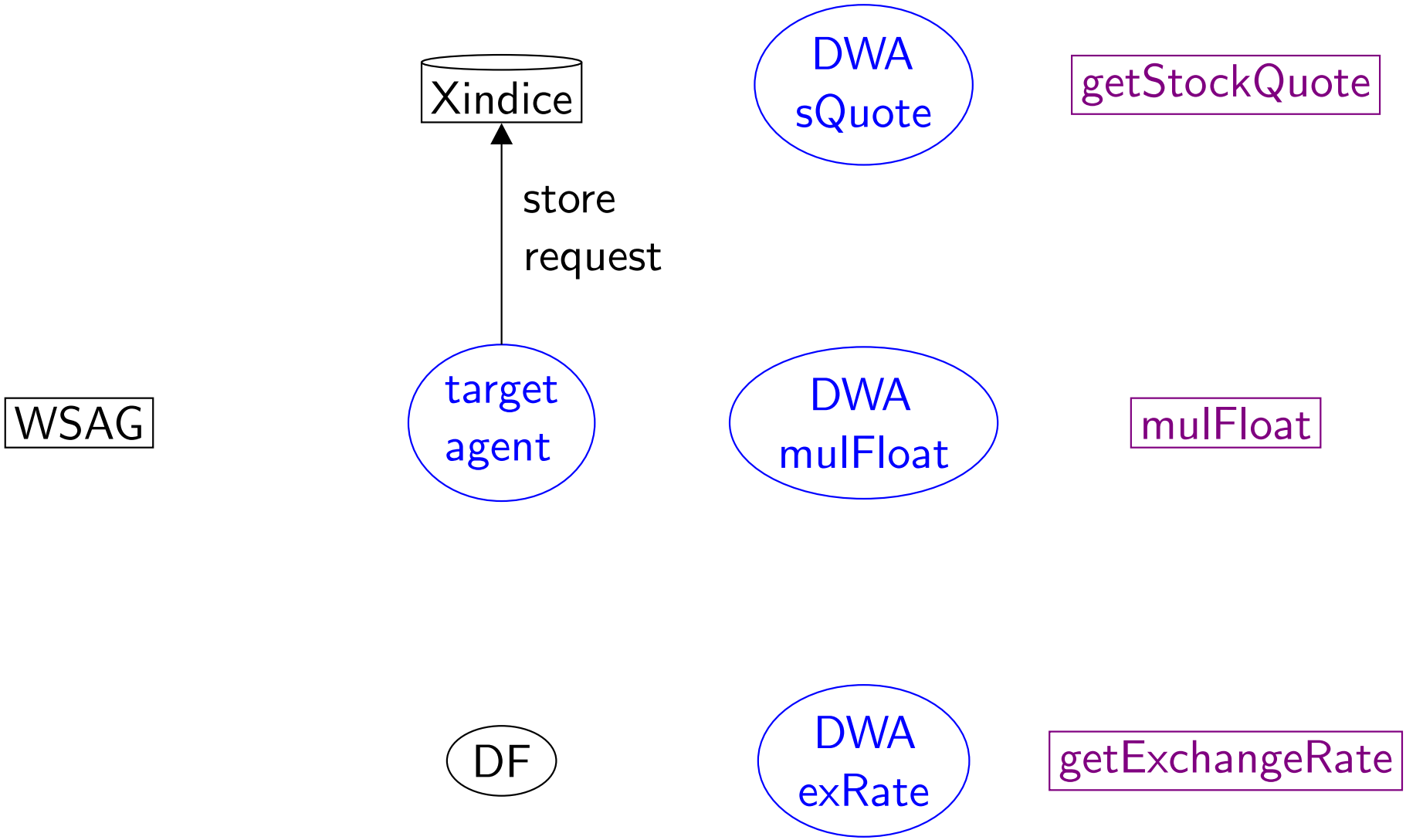
Multiagent Workflow Enactment Example



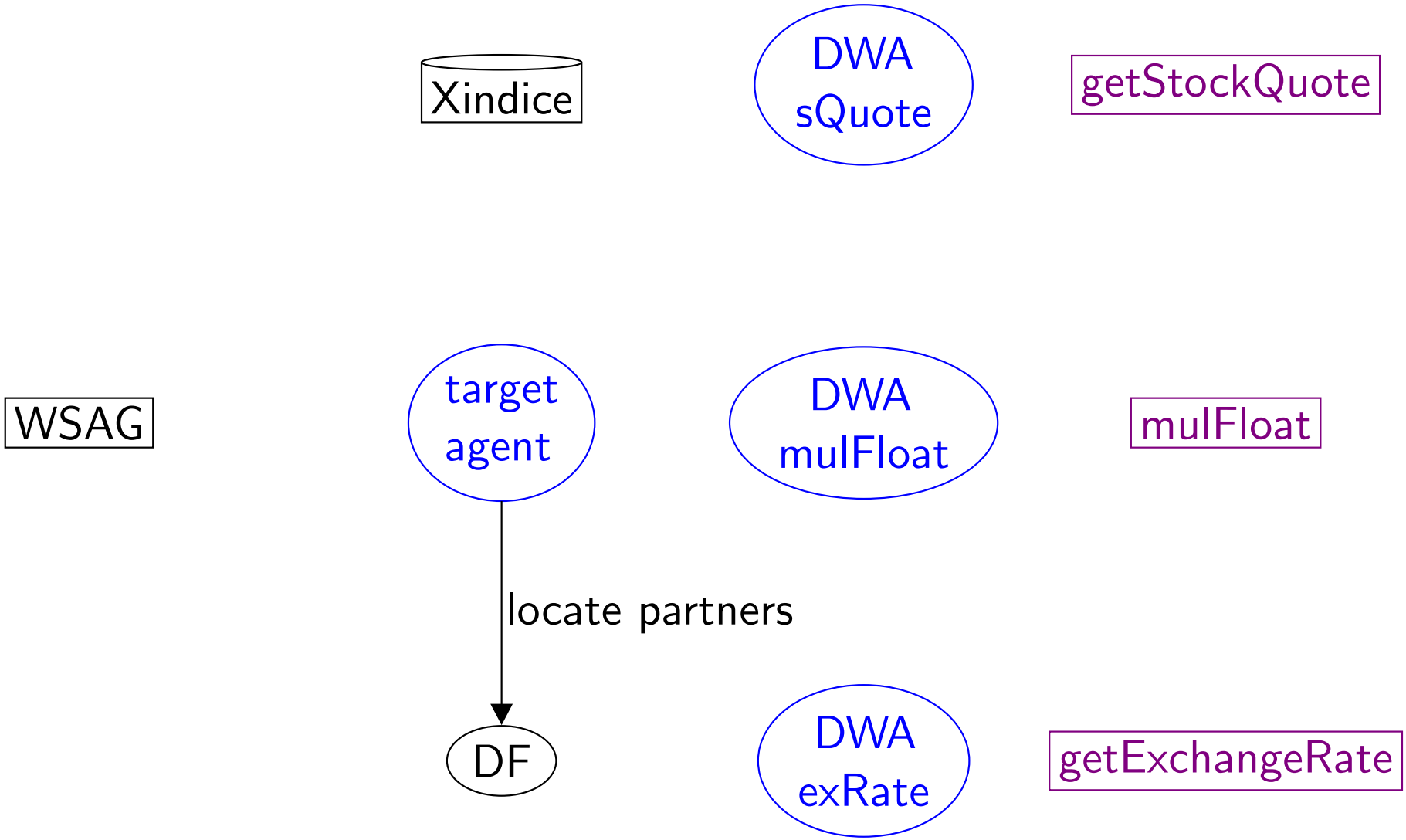
Multiagent Workflow Enactment Example



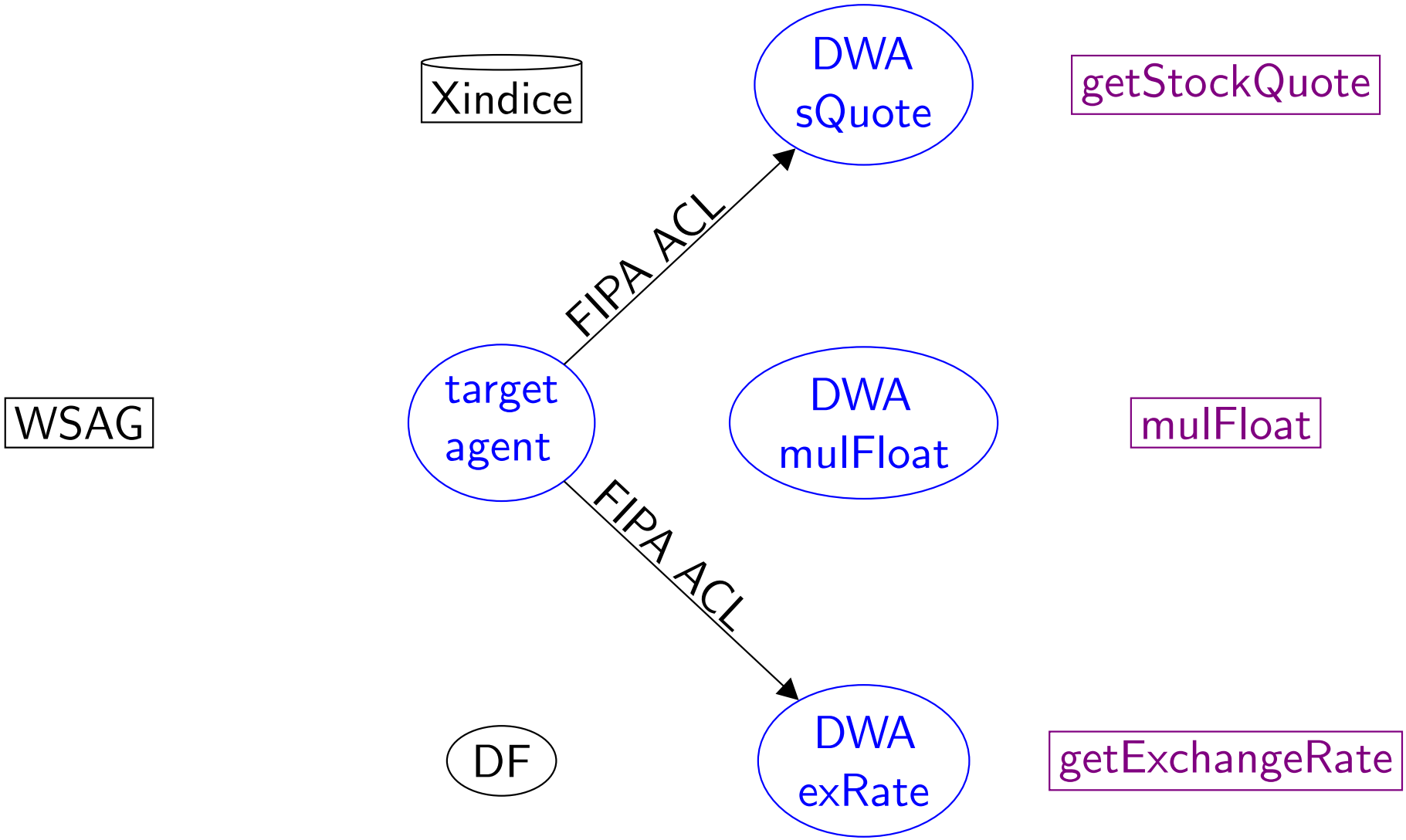
Multiagent Workflow Enactment Example



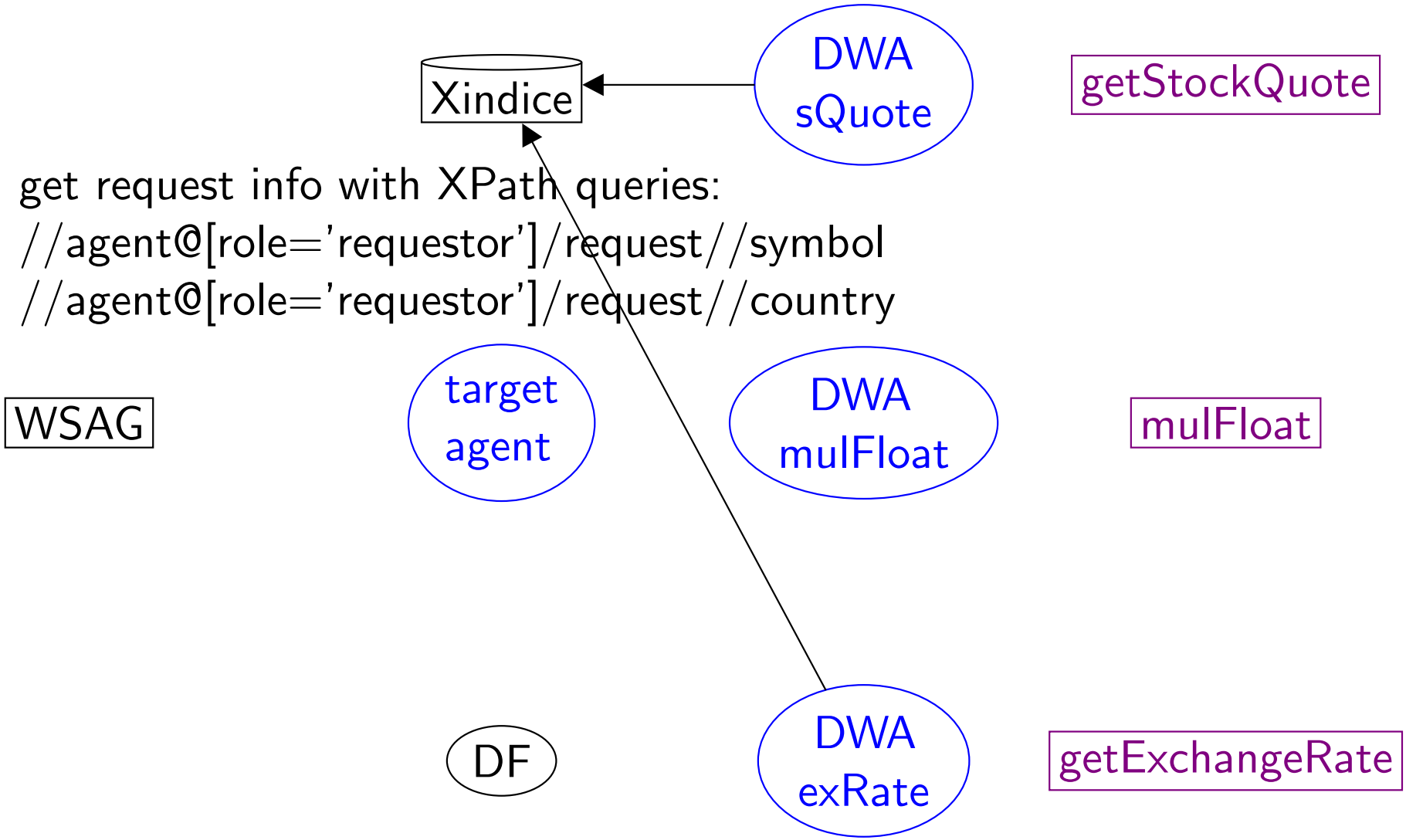
Multiagent Workflow Enactment Example



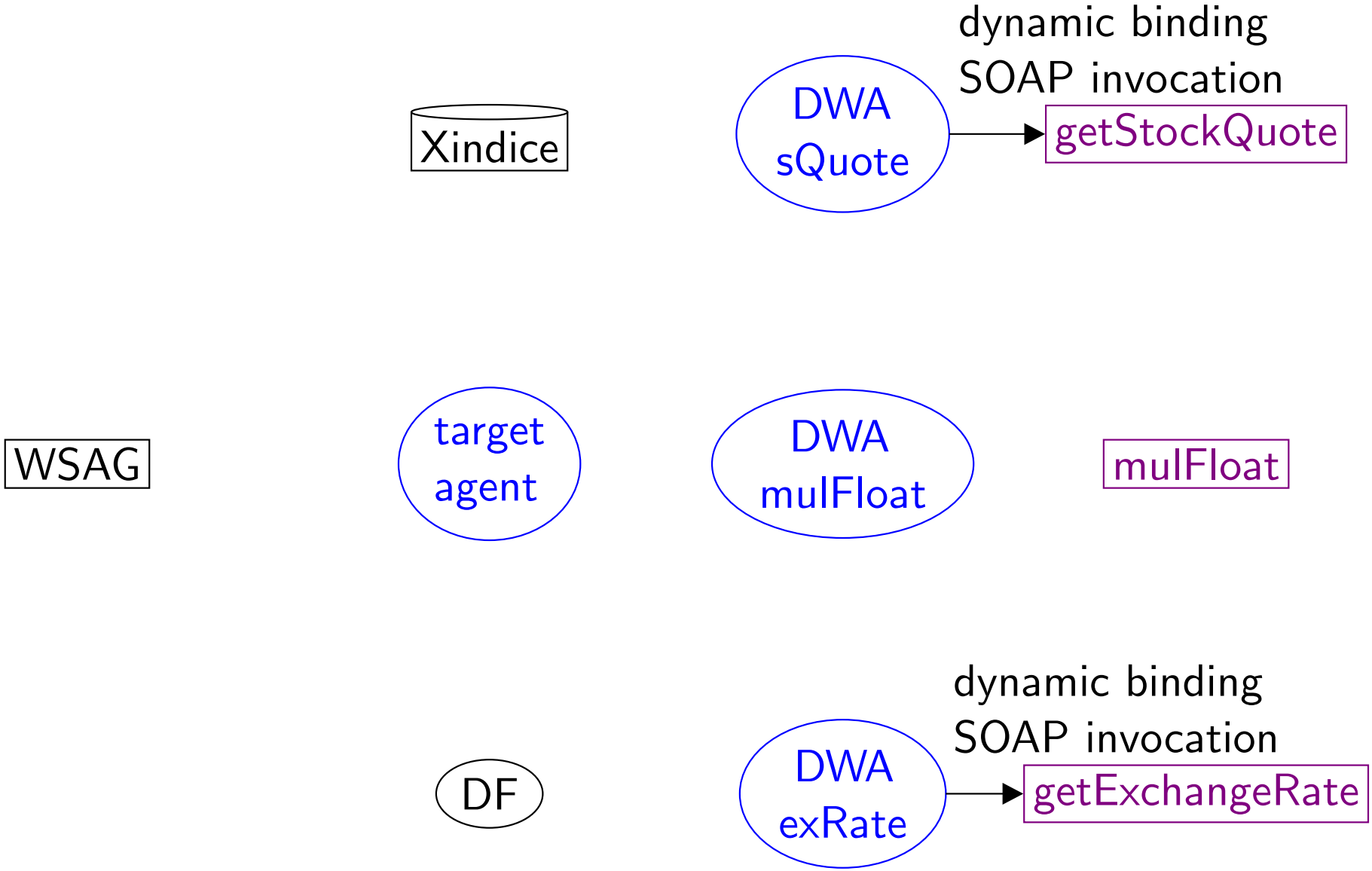
Multiagent Workflow Enactment Example



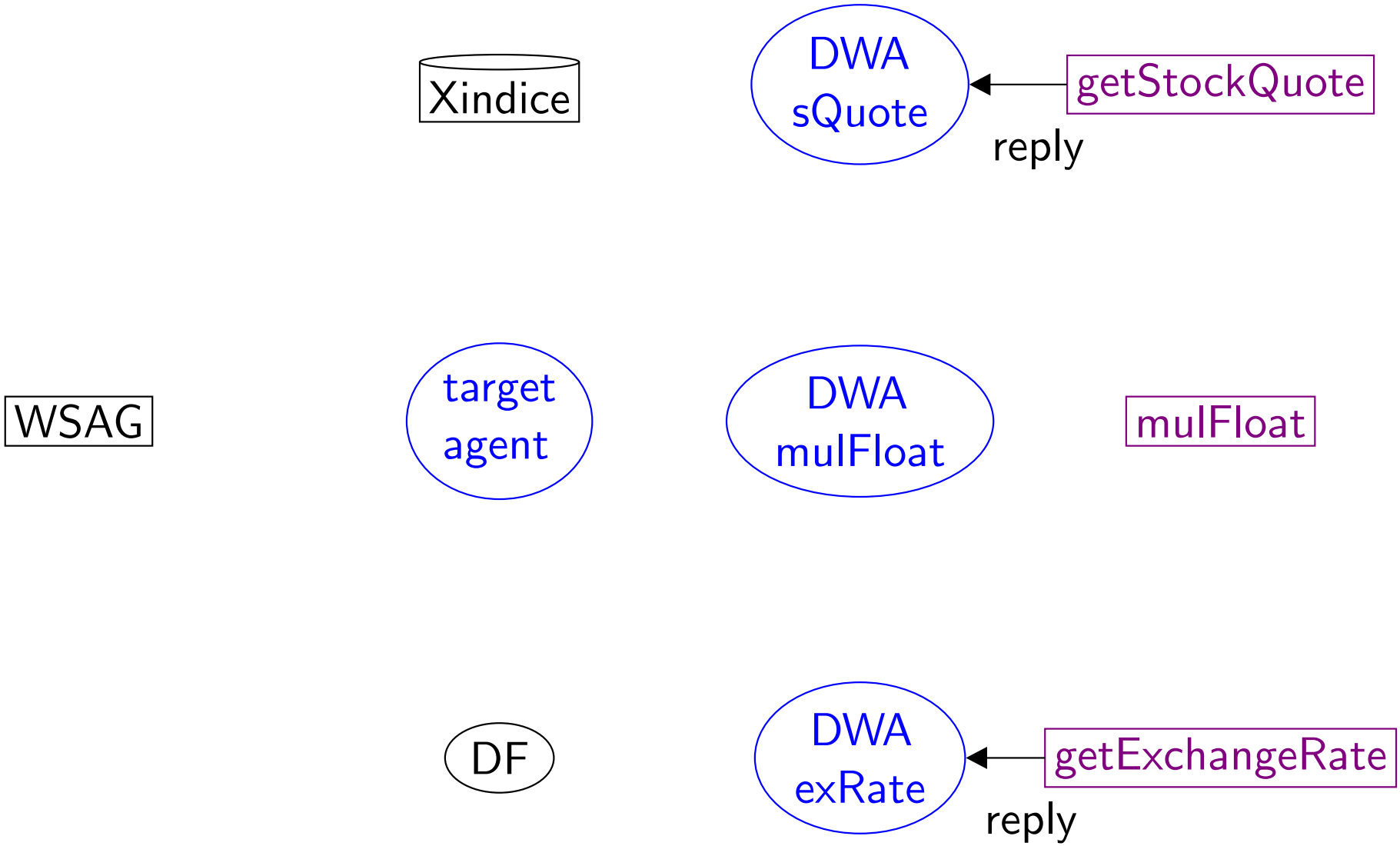
Multiagent Workflow Enactment Example



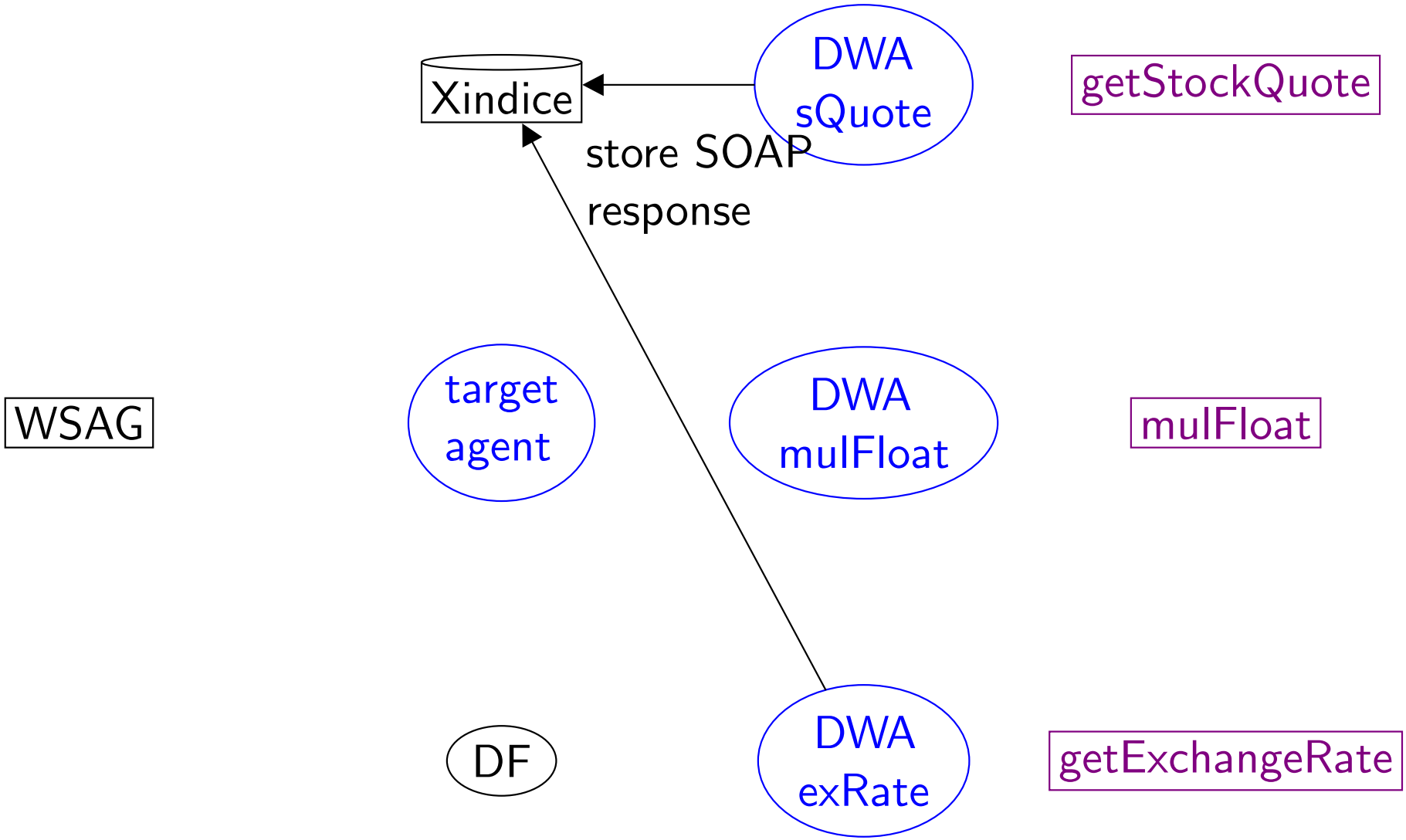
Multiagent Workflow Enactment Example



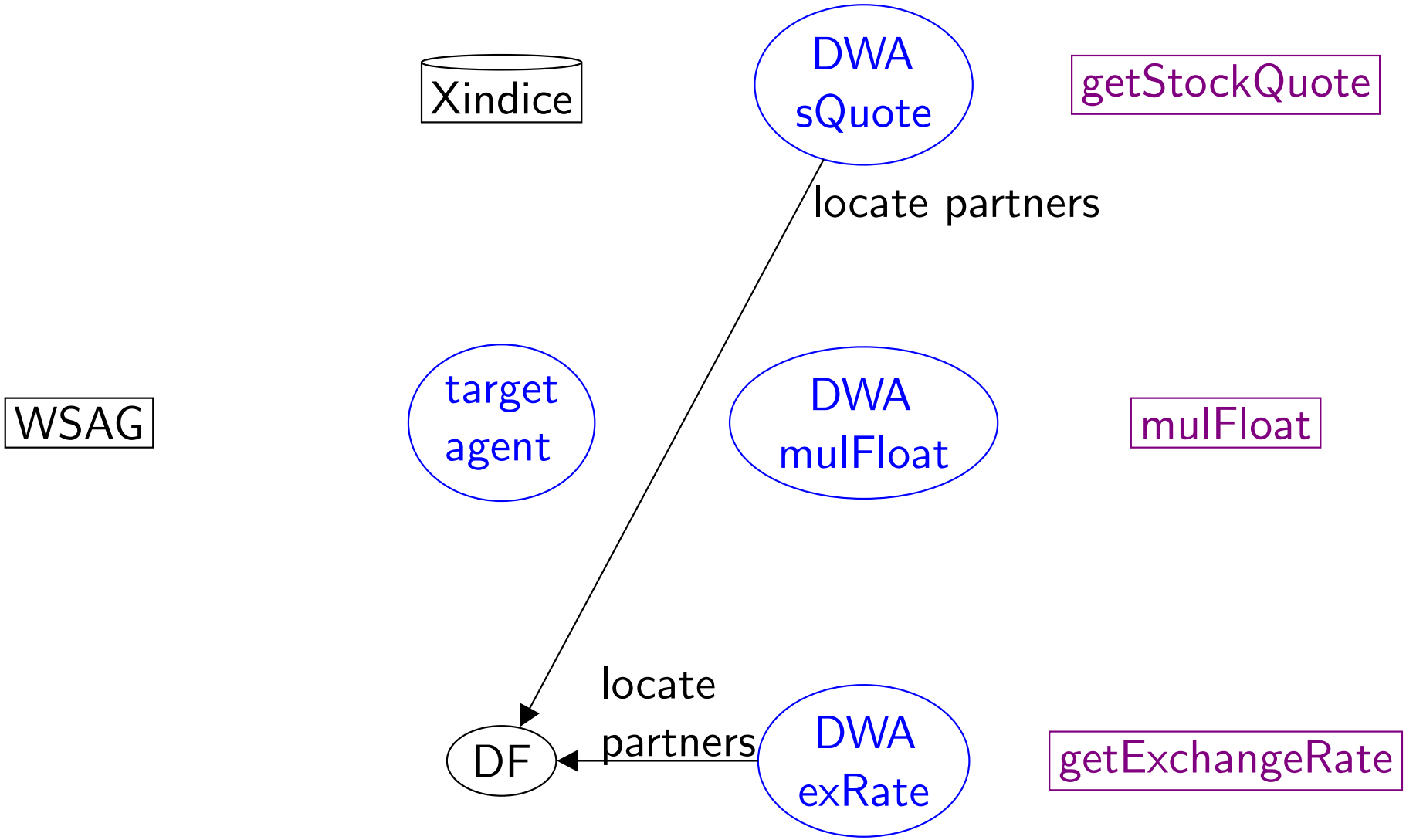
Multiagent Workflow Enactment Example



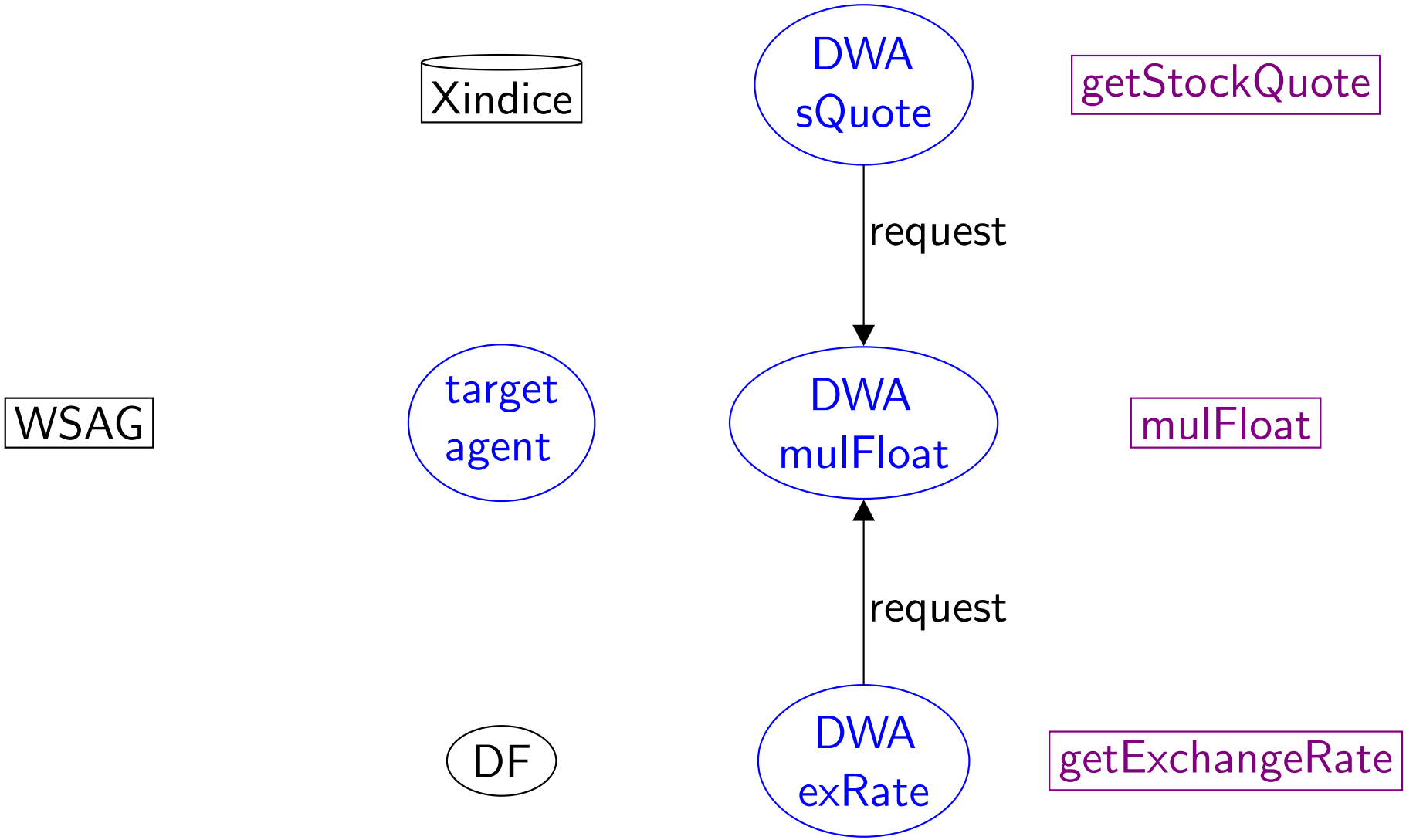
Multiagent Workflow Enactment Example



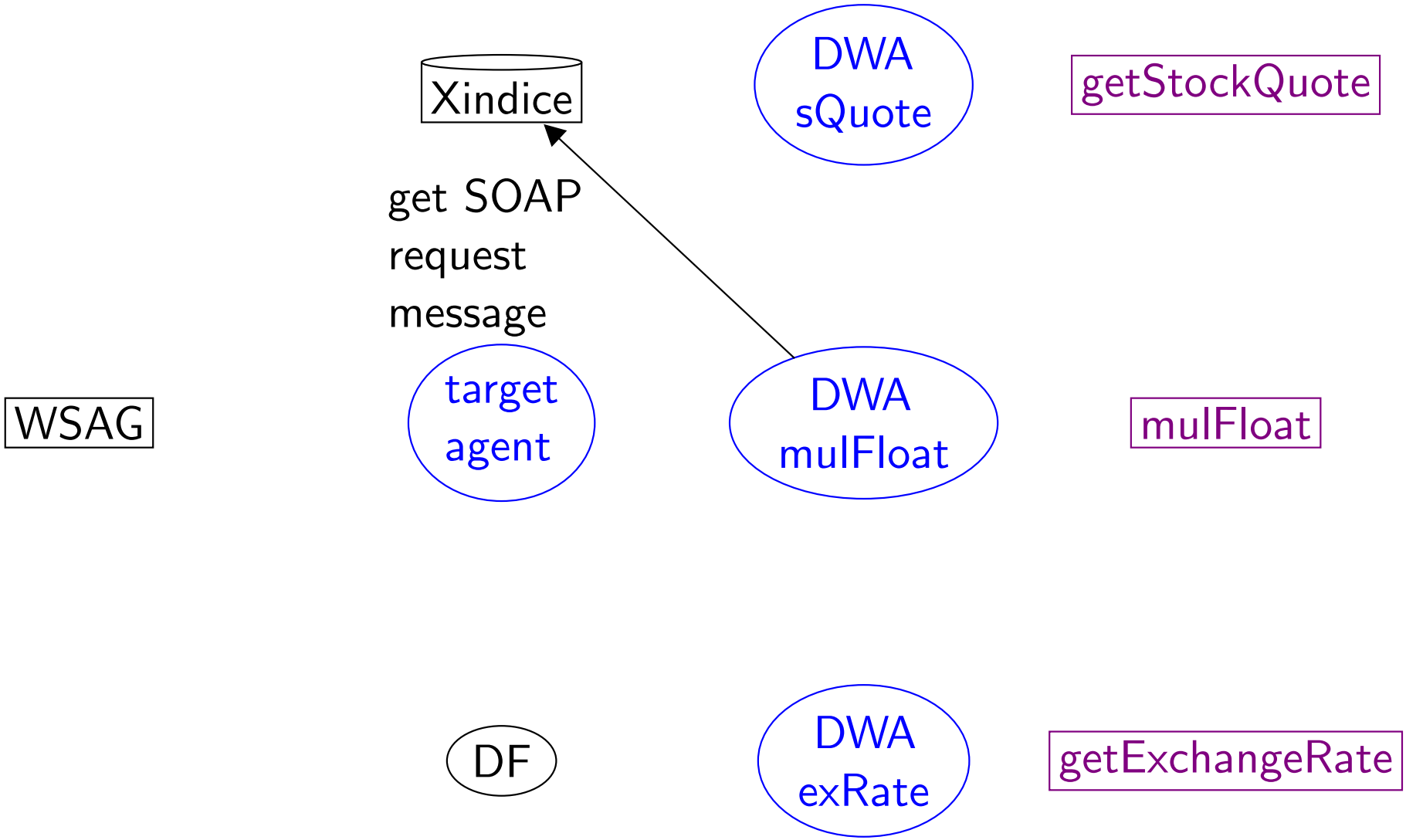
Multiagent Workflow Enactment Example



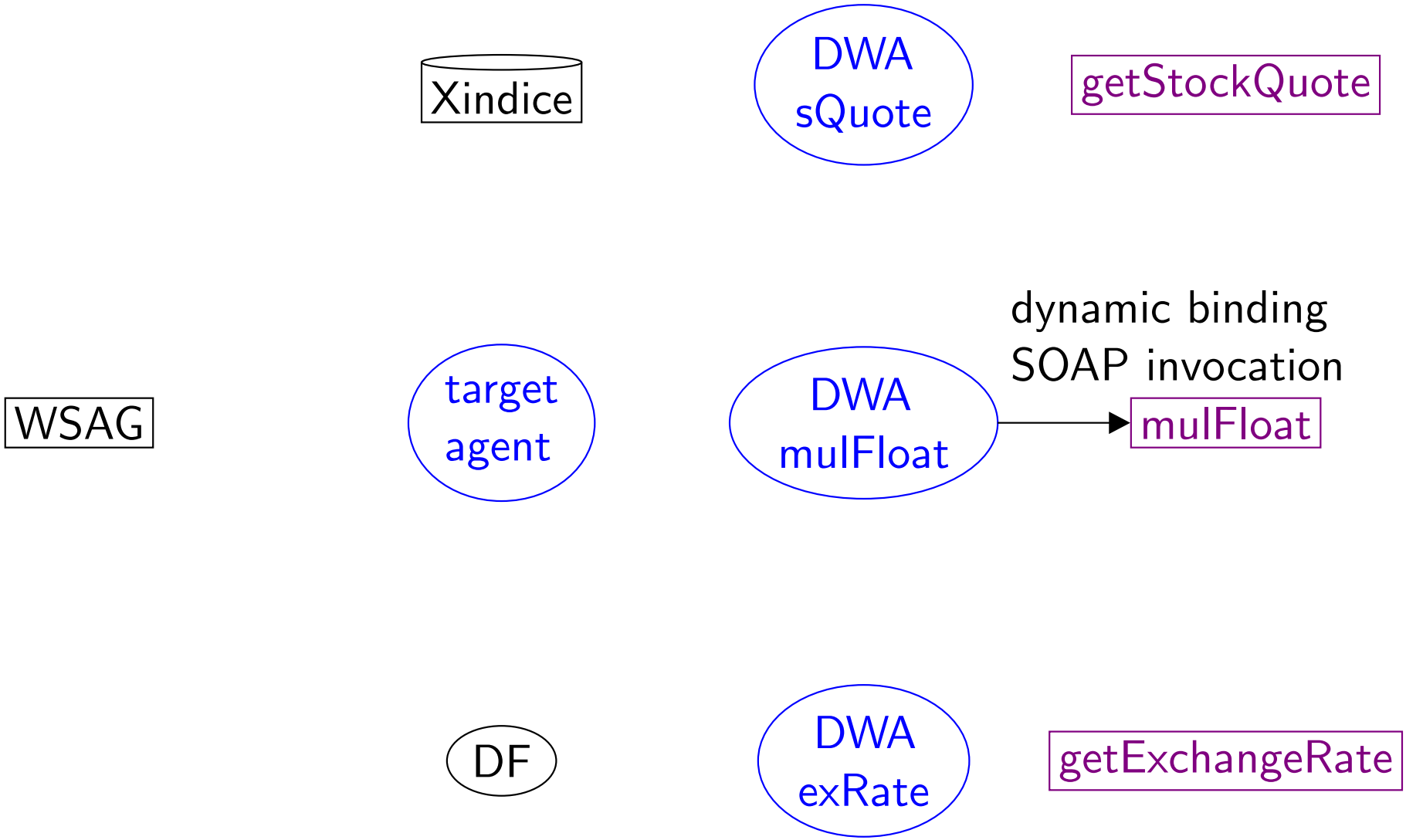
Multiagent Workflow Enactment Example



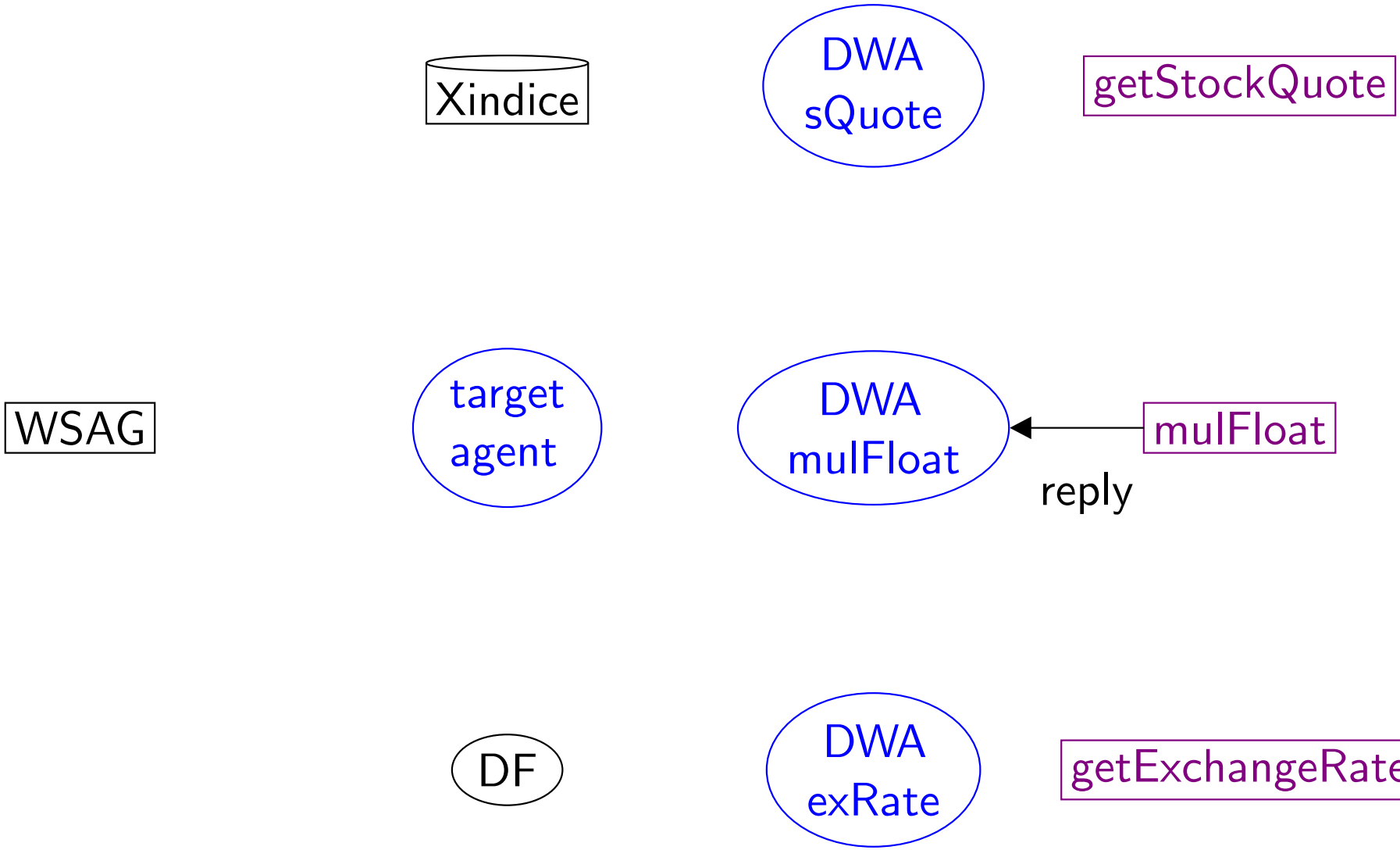
Multiagent Workflow Enactment Example



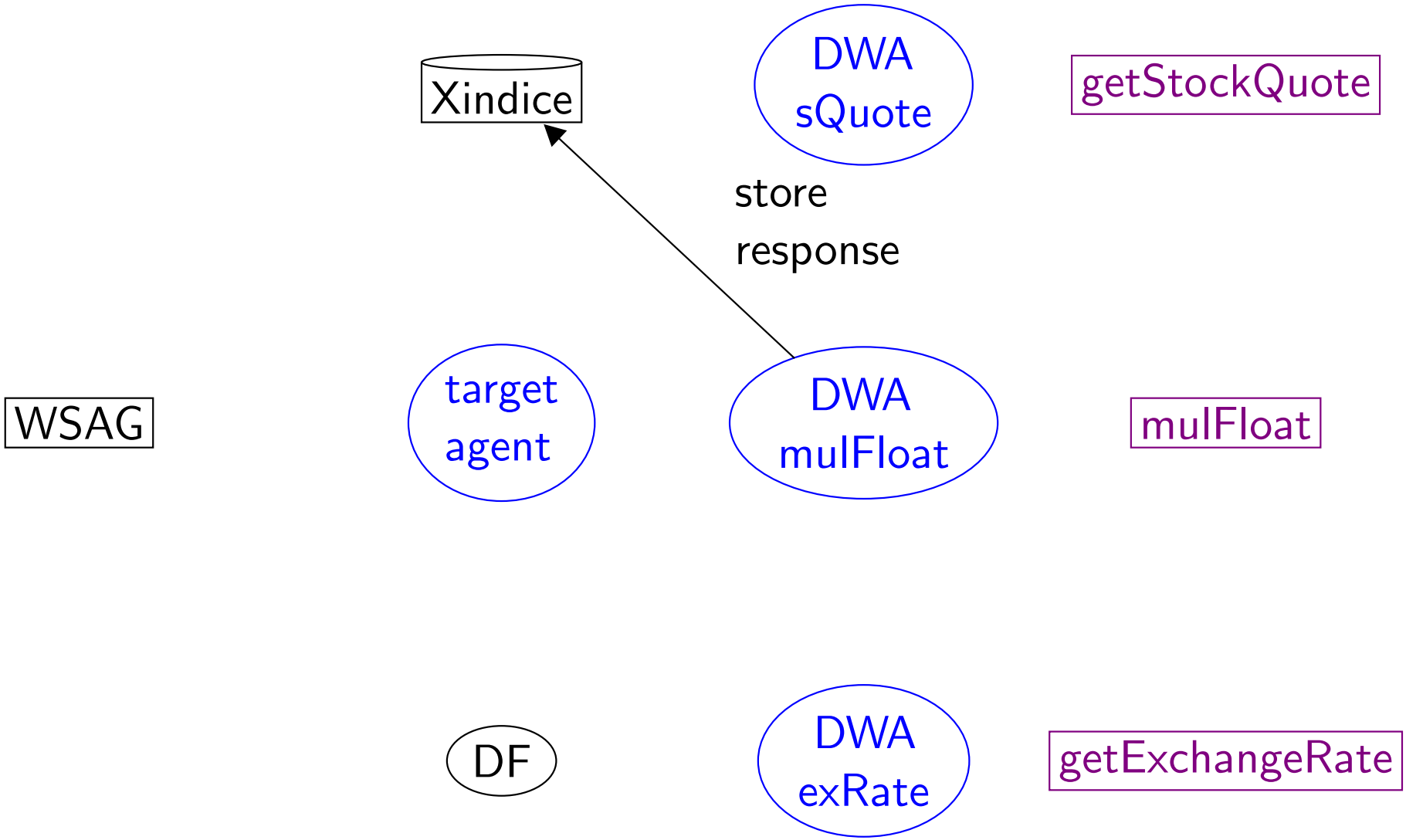
Multiagent Workflow Enactment Example



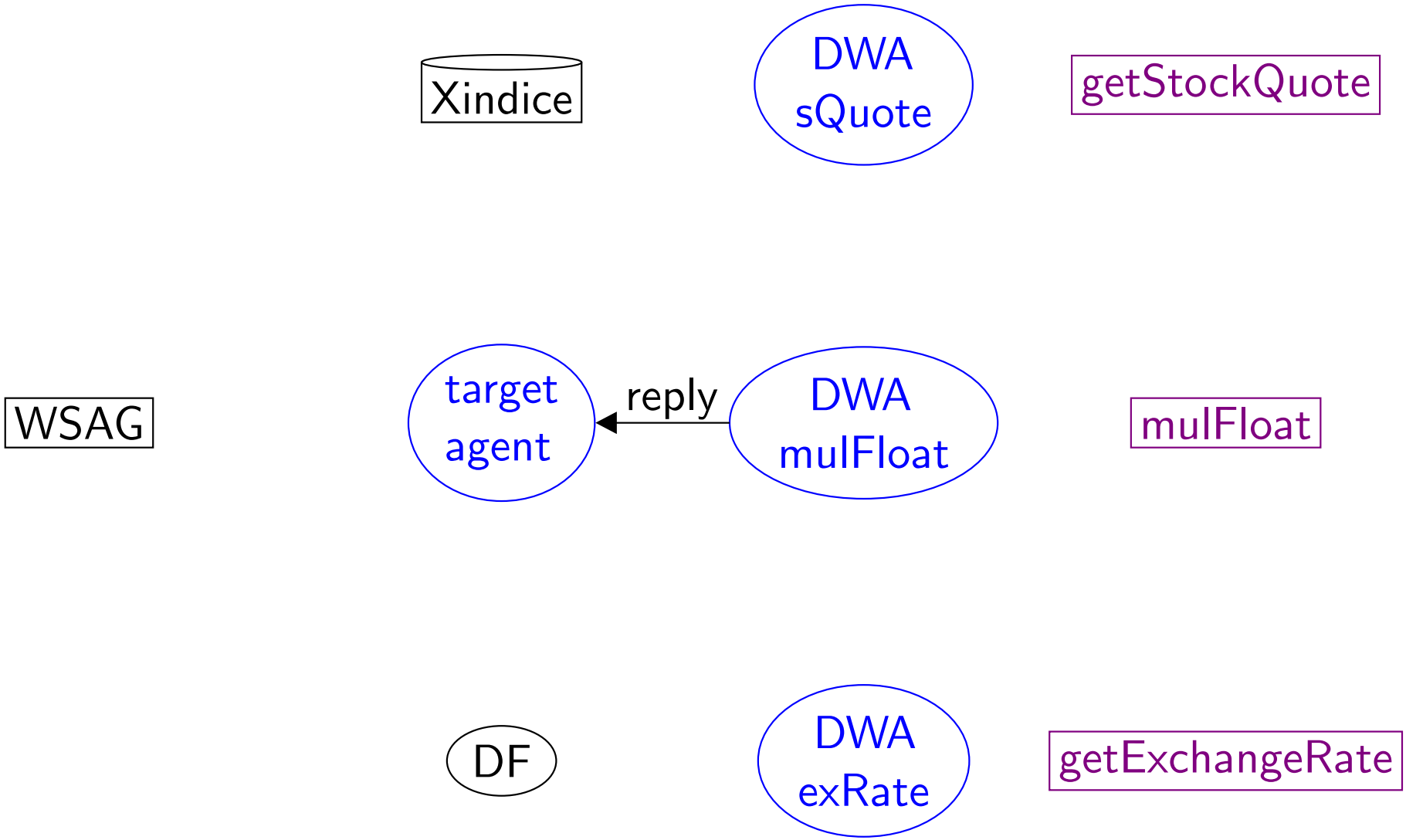
Multiagent Workflow Enactment Example



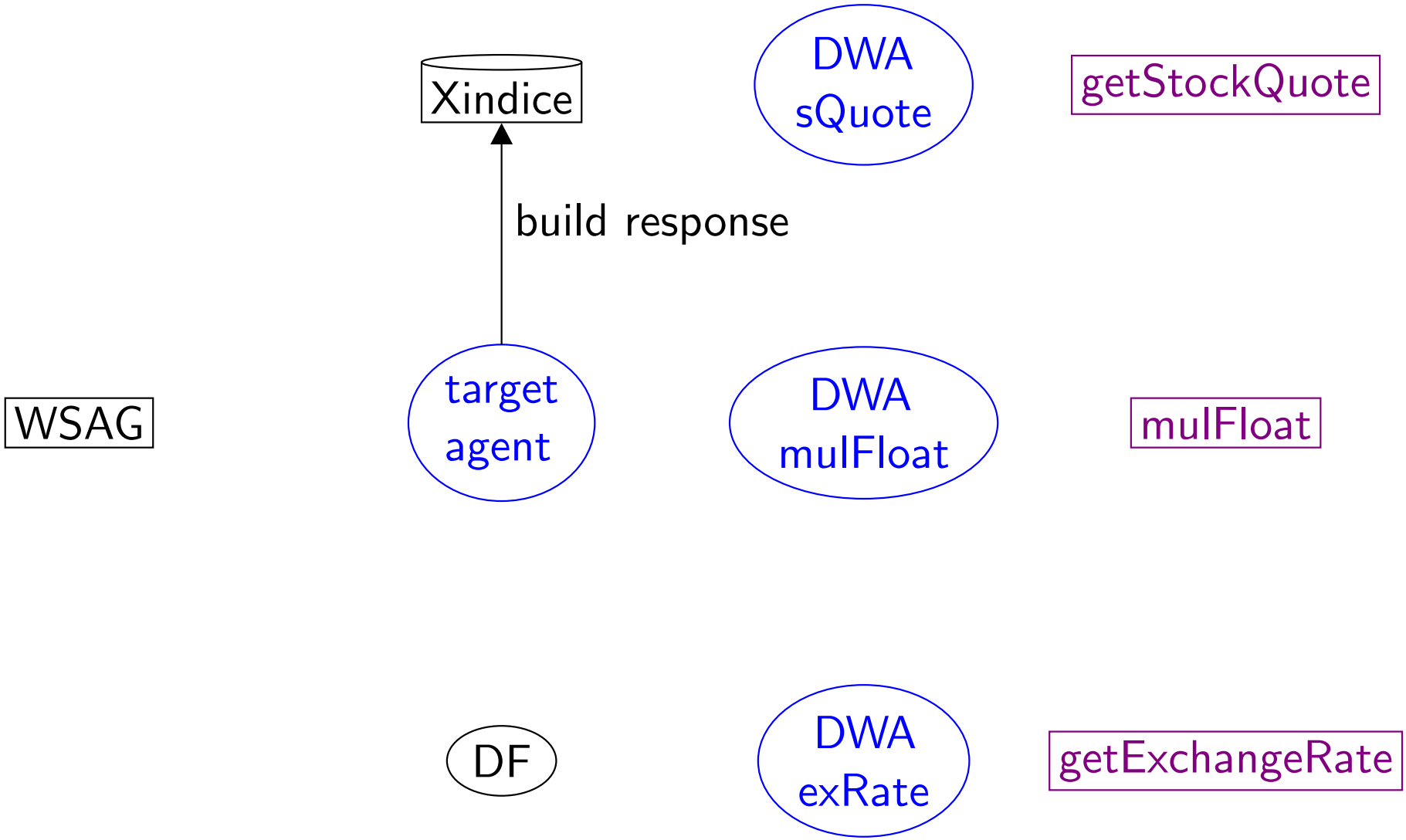
Multiagent Workflow Enactment Example



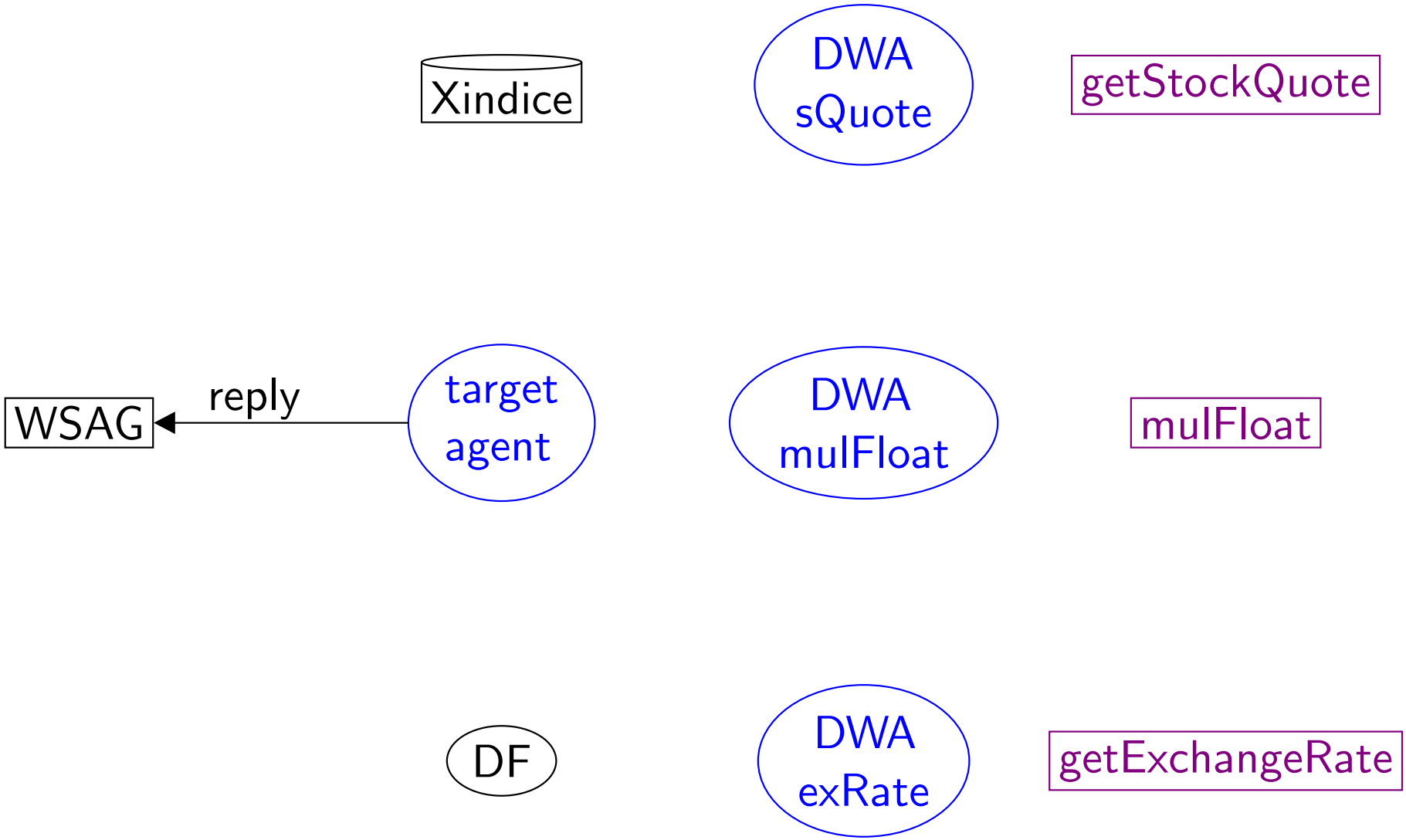
Multiagent Workflow Enactment Example



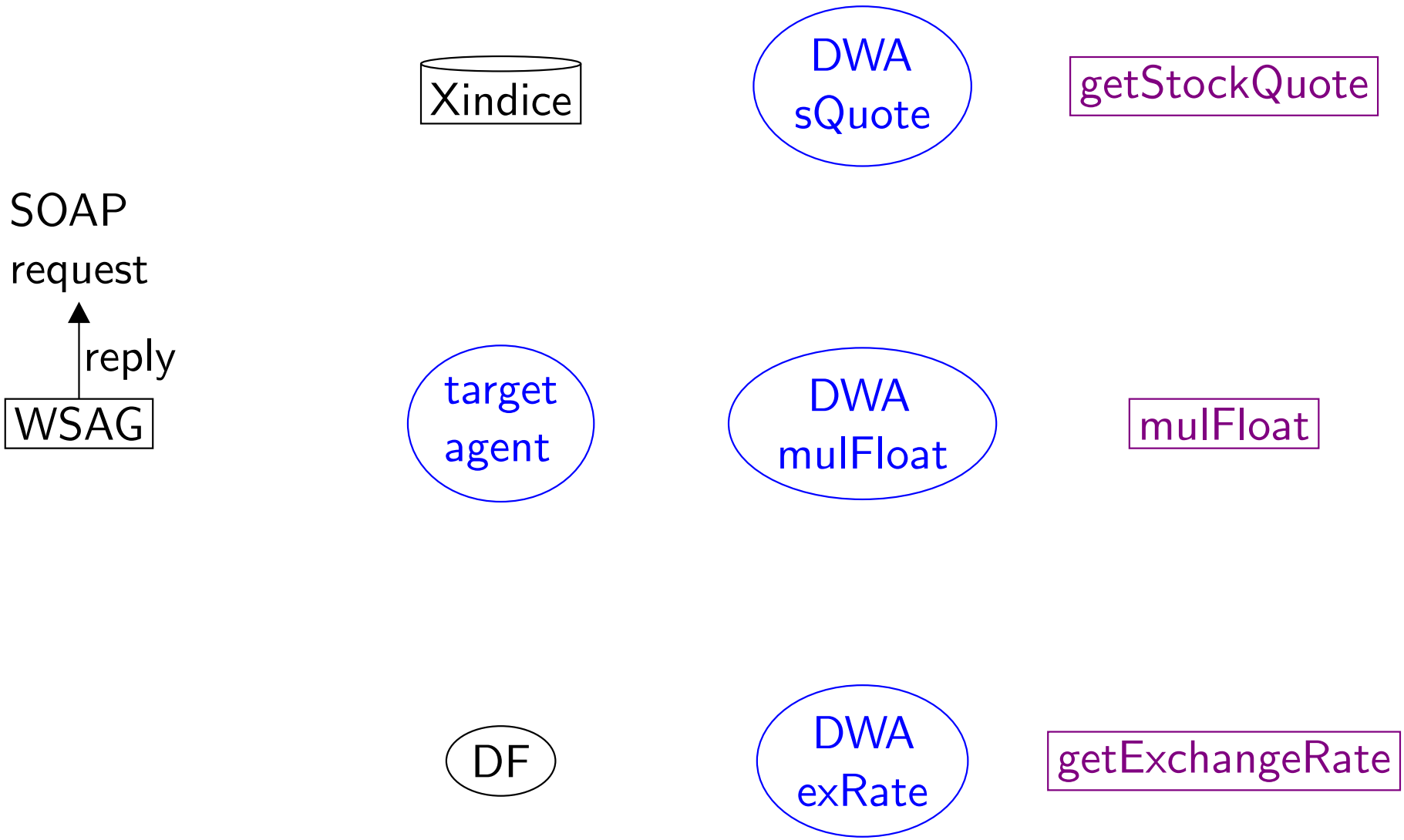
Multiagent Workflow Enactment Example



Multiagent Workflow Enactment Example



Multiagent Workflow Enactment Example



Lessons Learned

- It is possible and practical to use Web services as external behaviors, allowing for generic agent code.
- Not unmarshalling SOAP responses insulates code from differences in RPC versus DOC SOAP styles.
  - XML database is natural.
  - Allows for fully stubless dynamic invocation as no class is required to hold unmarshalled response.
  - Preserves namespace information which in the future will likely have semantic importance.
- DF useful for dynamic allocation.
- Hybrid coordination model proved to be flexible.
- WSAG allows WS clients to interact with the MAS transparently and enables MAS to be used as a sub-process within another workflow enacted by a commercial workflow engine.

Lessons Learned

- It is possible and practical to use Web services as external behaviors, allowing for generic agent code.
- Not unmarshalling SOAP responses insulates code from differences in RPC versus DOC SOAP styles.
  - XML database is natural.
  - Allows for fully stubless dynamic invocation as no class is required to hold unmarshalled response.
  - Preserves namespace information which in the future will likely have semantic importance.
- DF useful for dynamic allocation.
- Hybrid coordination model proved to be flexible.
- WSAG allows WS clients to interact with the MAS transparently and enables MAS to be used as a sub-process within another workflow enacted by a commercial workflow engine.

Lessons Learned

- It is possible and practical to use Web services as external behaviors, allowing for generic agent code.
- Not unmarshalling SOAP responses insulates code from differences in RPC versus DOC SOAP styles.
  - XML database is natural.
  - Allows for fully stubless dynamic invocation as no class is required to hold unmarshalled response.
  - Preserves namespace information which in the future will likely have semantic importance.
- DF useful for dynamic allocation.
- Hybrid coordination model proved to be flexible.
- WSAG allows WS clients to interact with the MAS transparently and enables MAS to be used as a sub-process within another workflow enacted by a commercial workflow engine.

Lessons Learned

- It is possible and practical to use Web services as external behaviors, allowing for generic agent code.
- Not unmarshalling SOAP responses insulates code from differences in RPC versus DOC SOAP styles.
  - XML database is natural.
  - Allows for fully stubless dynamic invocation as no class is required to hold unmarshalled response.
  - Preserves namespace information which in the future will likely have semantic importance.
- DF useful for dynamic allocation.
- Hybrid coordination model proved to be flexible.
- WSAG allows WS clients to interact with the MAS transparently and enables MAS to be used as a sub-process within another workflow enacted by a commercial workflow engine.

Lessons Learned

- It is possible and practical to use Web services as external behaviors, allowing for generic agent code.
- Not unmarshalling SOAP responses insulates code from differences in RPC versus DOC SOAP styles.
  - XML database is natural.
  - Allows for fully stubless dynamic invocation as no class is required to hold unmarshalled response.
  - Preserves namespace information which in the future will likely have semantic importance.
- DF useful for dynamic allocation.
- Hybrid coordination model proved to be flexible.
- WSAG allows WS clients to interact with the MAS transparently and enables MAS to be used as a sub-process within another workflow enacted by a commercial workflow engine.

Lessons Learned

- It is possible and practical to use Web services as external behaviors, allowing for generic agent code.
- Not unmarshalling SOAP responses insulates code from differences in RPC versus DOC SOAP styles.
  - XML database is natural.
  - Allows for fully stubless dynamic invocation as no class is required to hold unmarshalled response.
  - Preserves namespace information which in the future will likely have semantic importance.
- DF useful for dynamic allocation.
- Hybrid coordination model proved to be flexible.
- WSAG allows WS clients to interact with the MAS transparently and enables MAS to be used as a sub-process within another workflow enacted by a commercial workflow engine.

Lessons Learned

- It is possible and practical to use Web services as external behaviors, allowing for generic agent code.
- Not unmarshalling SOAP responses insulates code from differences in RPC versus DOC SOAP styles.
  - XML database is natural.
  - Allows for fully stubless dynamic invocation as no class is required to hold unmarshalled response.
  - Preserves namespace information which in the future will likely have semantic importance.
- DF useful for dynamic allocation.
- Hybrid coordination model proved to be flexible.
- WSAG allows WS clients to interact with the MAS transparently and enables MAS to be used as a sub-process within another workflow enacted by a commercial workflow engine.

Lessons Learned

- It is possible and practical to use Web services as external behaviors, allowing for generic agent code.
- Not unmarshalling SOAP responses insulates code from differences in RPC versus DOC SOAP styles.
  - XML database is natural.
  - Allows for fully stubless dynamic invocation as no class is required to hold unmarshalled response.
  - Preserves namespace information which in the future will likely have semantic importance.
- DF useful for dynamic allocation.
- Hybrid coordination model proved to be flexible.
- WSAG allows WS clients to interact with the MAS transparently and enables MAS to be used as a sub-process within another workflow enacted by a commercial workflow engine.

Future Work

- Implement `switch` and `pick` BPEL4WS constructs.
- Aggressive dynamic allocation.
- Integrate with Semantic Discovery Services.
- Workflow as skeletons for adaptive multiagent systems?