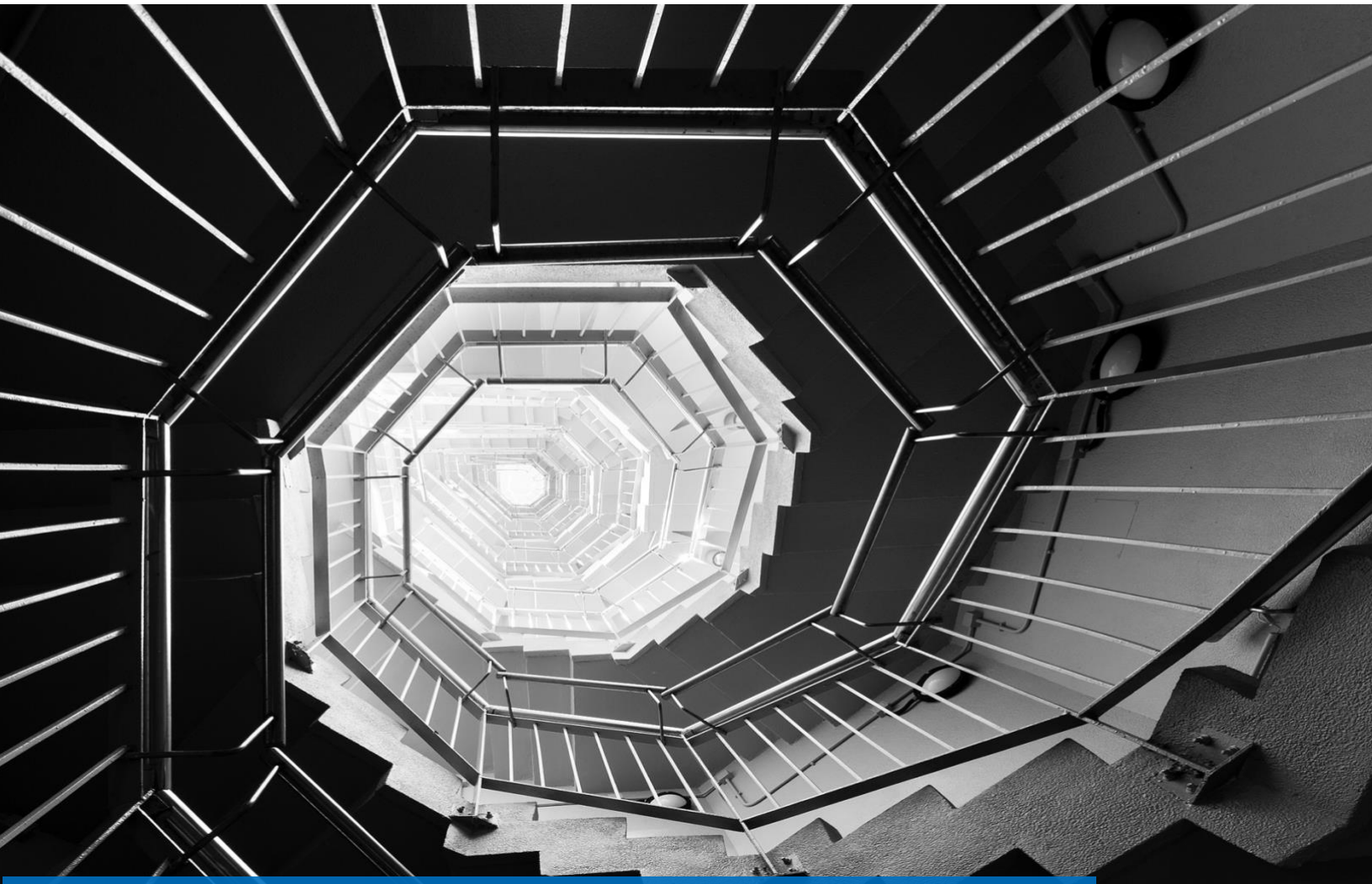

Building Cognitive Applications

Atlas Illustrations Reference

TABLE OF CONTENTS

Getting started

Welcome to the Atlas System (Atlas) Illustrations Reference.

Atlas offers multiple co-dependent components (capabilities) available through C++ APIs.

Atlas system documentation is intended for developers and subject matter experts seeking to improve cognitive applications, services, and experiences.

To learn how to develop cognitive applications using Atlas, we recommend starting with the document “Atlas System Guide” followed by the “Atlas Process Guide” which is also part of the Atlas Development Kit.

Atlas

System Components

Atlas system components

Atlas is composed of multiple co-dependent components for learning, understanding, reasoning and communicating knowledge.

Atlas components are available through C++ APIs.

I. Knowledge Representation

The topology of a cognitive environment (CE) is organized into **Domains**, **Zones**, **Knowledge Bases**, and Knowledge **Simulations**.

Domains contain knowledge bases, zones, and simulations.

	Domain
	Knowledge Base
	Simulation
	Zone

Fig 1. Knowledge representation components.

II. Knowledge Acquisition

The main structuring element of a CE is a **concept**.

Concepts are made out of **declarations**, **definitions**, and **interpretations**.

Concepts are learned through **manifests**.

	Concept
	Declaration
	Definition
	Interpertation
	Manifest

Fig 2. Knowledge acquisition components.

III. Knowledge Handling

Concept instances represent the current state of the cognitive system. **Instantiators** create and destroy **instances**.

When an instance is labeled, it becomes a **reference**.

Multiple references can be combined into a **group.**

Internal simulation instances are called **attributes**.

	Instantiator
	Instance
	Reference
	Group
	Group member
	Attribute

Fig 3. Knowledge handling components.

IV. Knowledge Access

Any instance can have one or multiple names or **labels**.

Conversation between agents is performed through **expressions**.

Some conversations have extra payloads known as **attachments**.

Another more structured form of conversation is the **directive**.

Scripts are files with one or multiple expressions.

Ledgers are expression recorders and players.

Titan Exchange is the ecosystem where one of multiple cognitive agents can semantically search for information and collaborate on solutions through recursive interaction and delegation.

	Label
	Expression
	Attachment
	Directive
	Script
	Ledger
	Titan Exchange

Fig 4. Knowledge access components.

V. Knowledge Execution

Execution in CE is achieved through Atlas API calls.

Interface methods are supplied by Atlas to the developer to manage the cognitive environment.

User manager methods are supplied by the developer to Atlas to regulate the component being managed.

State axiom methods simulate the **Be** aspects of a defined concept.

State change axiom methods simulate the **Do** aspects of a defined concept.

State relation axiom methods simulate the **Have** aspects of a defined concept.

	Titan interface method
	User manager method
	State axiom method: Be
	State change axiom method: Do
	State relation axiom method: Have

Fig 5. Knowledge execution components.

VI. Knowledge Interaction

Activity in CE is performed either by a *user* or an *agent*.

Users in this document represent a human entity. Atlas is a purely abstract cognitive agent; however, it can clone itself in one or multiple domains.

An Atlas **clone** is a cognitive agent that can assume different identities known as **context agents**.

A context agent that has dominion over a component such as a domain or a zone is called a **managing agent**.

	User
	Atlas Clone
	Context Agent
	Managing Agent

Fig 6. Knowledge interaction components.

VII. Knowledge Exchange

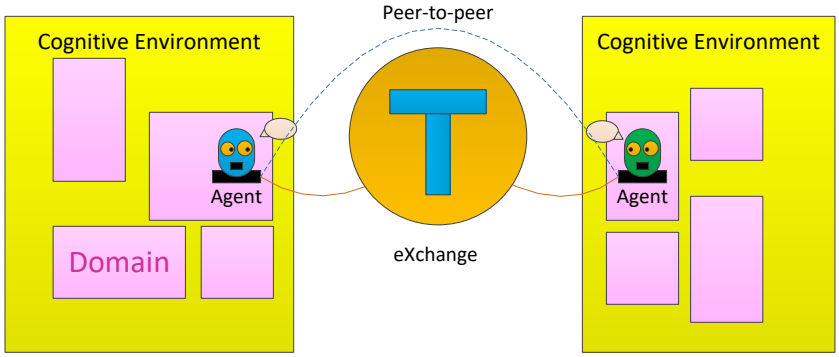

Fig 7. High-level overview of a Titan eXchange infrastructure featuring two agents, each managing their own respective domains: Communication between the agents is facilitated through a multi-domain networked cognitive environment.

Atlas
System Interactions

Atlas System Overview

I. Atlas System

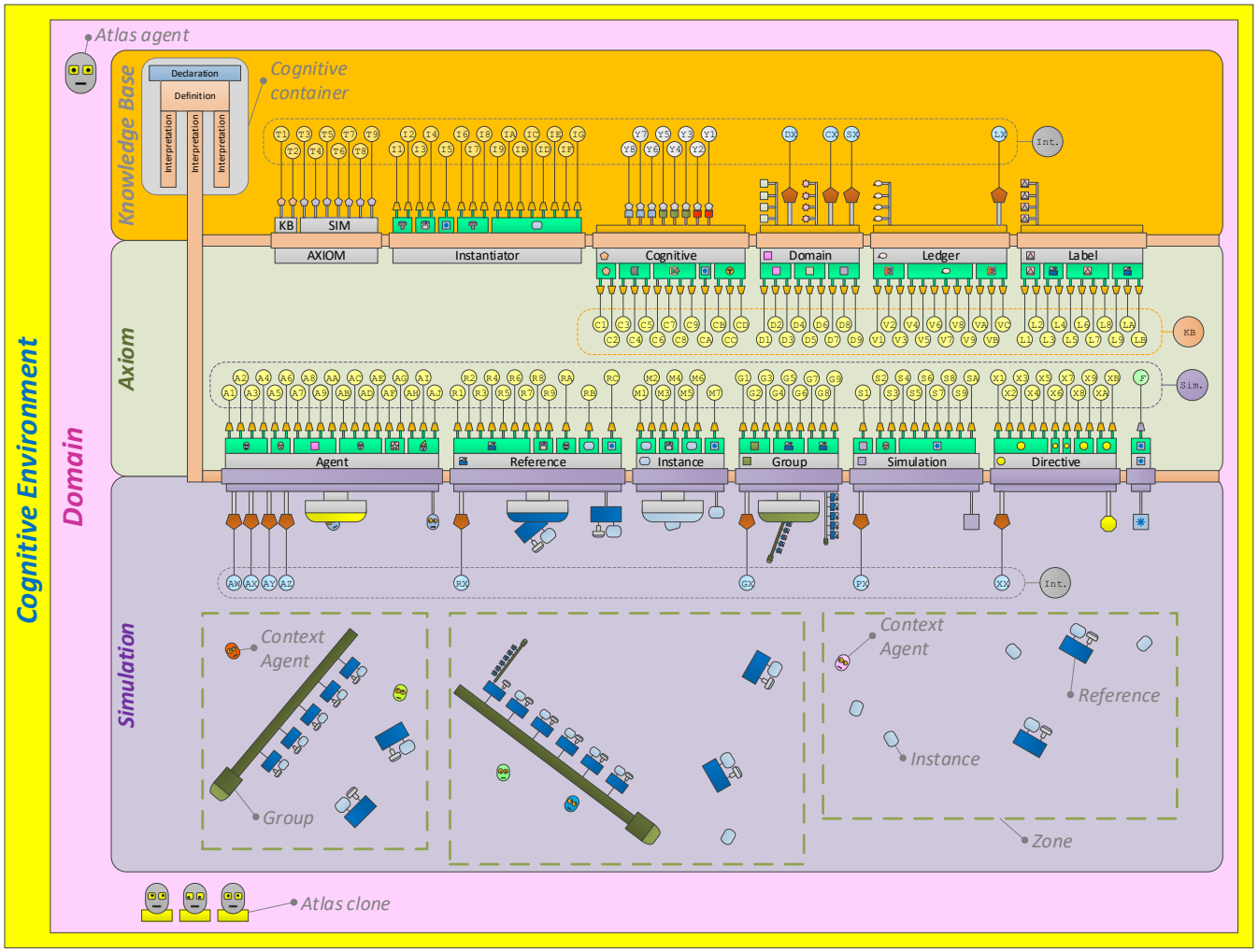

Fig A1. System Overview

II. Atlas Environment Interaction

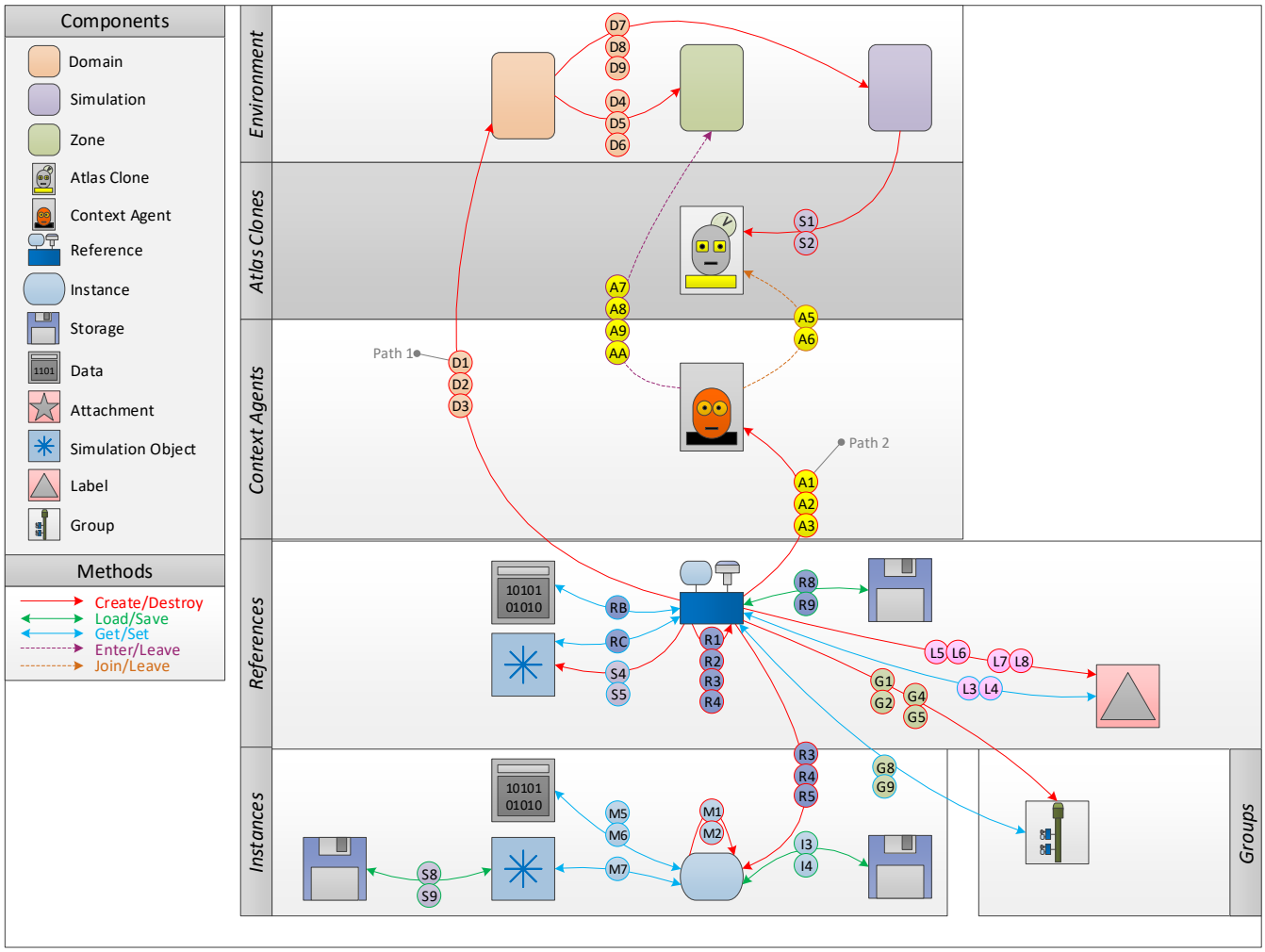

Fig A2. Environment Interaction

III. Atlas Payload Interaction

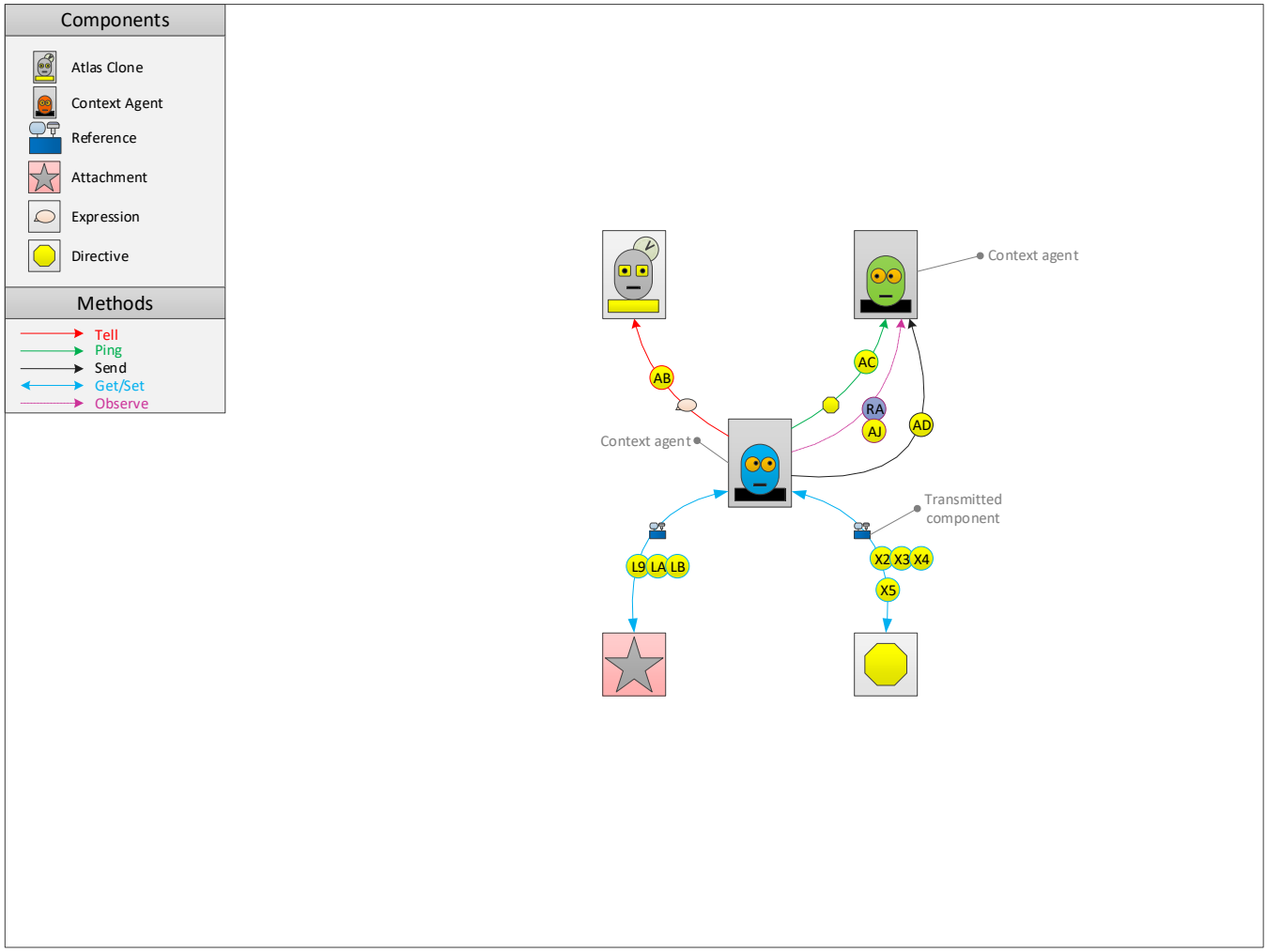

Fig A3. Atlas Payload Interaction

Atlas system callouts:

I. Concept Call Outs:

Annotation:

In the overview diagram, the concept API methods are confined to the following delimiter.

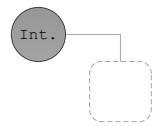

Fig 8. Concept API methods section delimiter.

Concept API Interfaces:

Axiom Manager	T1	KB_Attach
	T2	KB_Detach
	T3	Sim_Attach
	T4	Sim_Detach
	T5	Sim_Link
	T6	Sim_Unlink
	T7	Sim_Observe
	T8	Sim_Hear
	T9	Sim_Receive
Instance Manager	I1	Sim_Create
	I2	Sim_Destroy
	I3	Sim_Load
	I4	Sim_Save
	I5	Sim_GetSimulationObject
	I6	Sim_GetSub
	I7	Sim_GetSize
	I8	Sim_GetValue
	I9	Sim_SetValue
	IA	Sim_SetElementCount
	IB	Sim_GetElementCount
	IC	Sim_GetElement
	ID	Sim_GetFirstElement
	IE	Sim_GetNextElement
	IF	Sim_AddElement
	IG	Sim_RemoveElement

Fig 9. Common concept API methods: These virtual methods are inherited from AtlasAxiom and expanded by the developer to support the functionality requested by Atlas from the concept interpretation.

II. Cognitive Call Outs:

Annotations:

In the overview diagram, the Cognitive API methods are confined to the following delimiter.

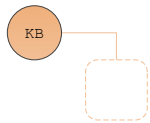

Fig 10. Knowledge Base methods section delimiter.

Cognitive API Interfaces

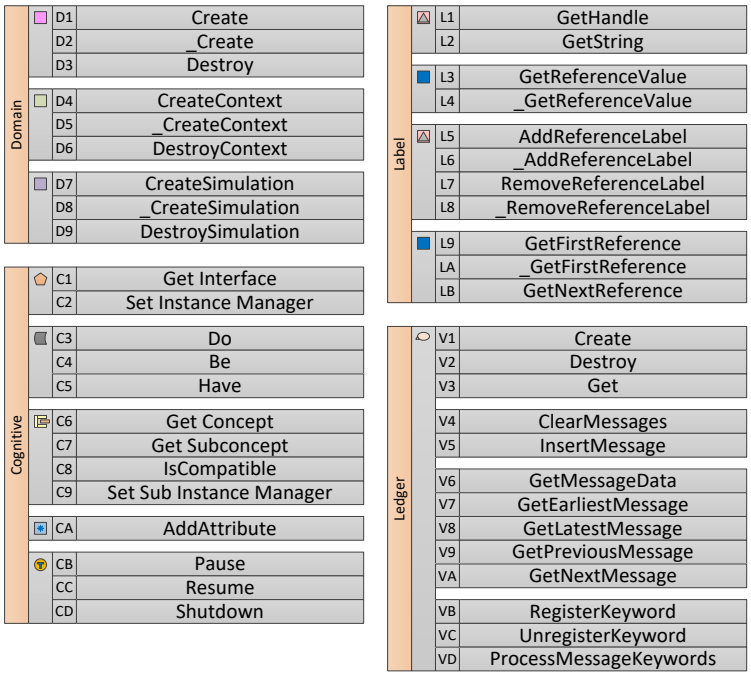

Fig 11. Callouts for the knowledge base methods and components.

III. Simulation Call Outs:

Annotation:

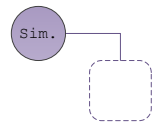

Fig 12. Simulation API methods

Simulation API Interfaces:

Simulation	S1	GetTime
	S2	Create*Atlas*Clone
	S3	Destroy*Atlas*Clone
	S4	CreateSimulationObject
	S5	DestroySimulationObject
	S6	SetSimulationObjectParent
	S7	LoadSimulationObject
	S8	SaveSimulationObject
	S9	ActivateSimulationObject
	SA	DeactiveSimulationObject
Agent	A1	Create
	A2	_Create
	A3	Destroy
	A4	RegisterManager
	A5	Join*Atlas*Clone
	A6	Leave*Atlas*Clone
	A7	EnterZone
	A8	_EnterZone
	A9	ExitZone
	AA	_ExitZone
	AB	Tell
	AC	Ping
	AD	Transmit
	AE	ExecuteScript
	AF	GetAttachment
	AG	_GetAttachment
	AH	RegisterExpert
	AI	UnregisterExpert
	AJ	Observe
Instance	M1	Create
	M2	Destroy
	M3	Load
	M4	Save
	M5	GetValue
	M6	SetValue
	M7	GetSimulationObject

Reference	R1	Add
	R2	_Add
	R3	Create
	R4	_Create
	R5	Destroy
	R6	RegisterManager
	R7	GetPublicHandle
	R8	Load
	R9	Save
	RA	Observe
	RB	GetInfo
	RC	GetTransform
Group	G1	Create
	G2	Destroy
	G3	Lock
	G4	AddMember
	G5	RemoveMember
	G6	Clear
	G7	GetMemberCount
	G8	GetFirst
	G9	GetNext
Directive	X1	GetNumComponents
	X2	GetComponent
	X3	GetNextComponent
	X4	GetComponentByName
	X5	RephraseComponent
	X6	GetAction
	X7	GetState
	X8	SetComponentHandshake
	X9	ClearHandshakes
	XA	Remember
	XB	Forget

Fig 13. Callouts for the simulation methods and components.