UCAIug OpenADR Taskforce Face to Face Meeting

May 4th – May 6th, 2010

3 Day Agenda

- Tuesday, May 4, 3:30 -5:30
  - Overview of existing 1.0 version of OpenADR specification and its data models, etc.
- Wedesday, May 5, 10:30-12:00
  - Confirm details of scope for SRS
  - Discussion of general inter-domain interaction models to be used in SRS
    - Which entities?
    - Type of interactions that are in scope of SRS?
- Wednesday, May 5, 3:30-5:30
  - Utility enterprise integration and harmonization issues to be addressed in SRS
    - CIM, AMI-ENT, OpenADE, SEP
- Thursday, May 6, 8:00-10:00
  - Work on SRS document itself.
    - Review current draft, identify areas of focus

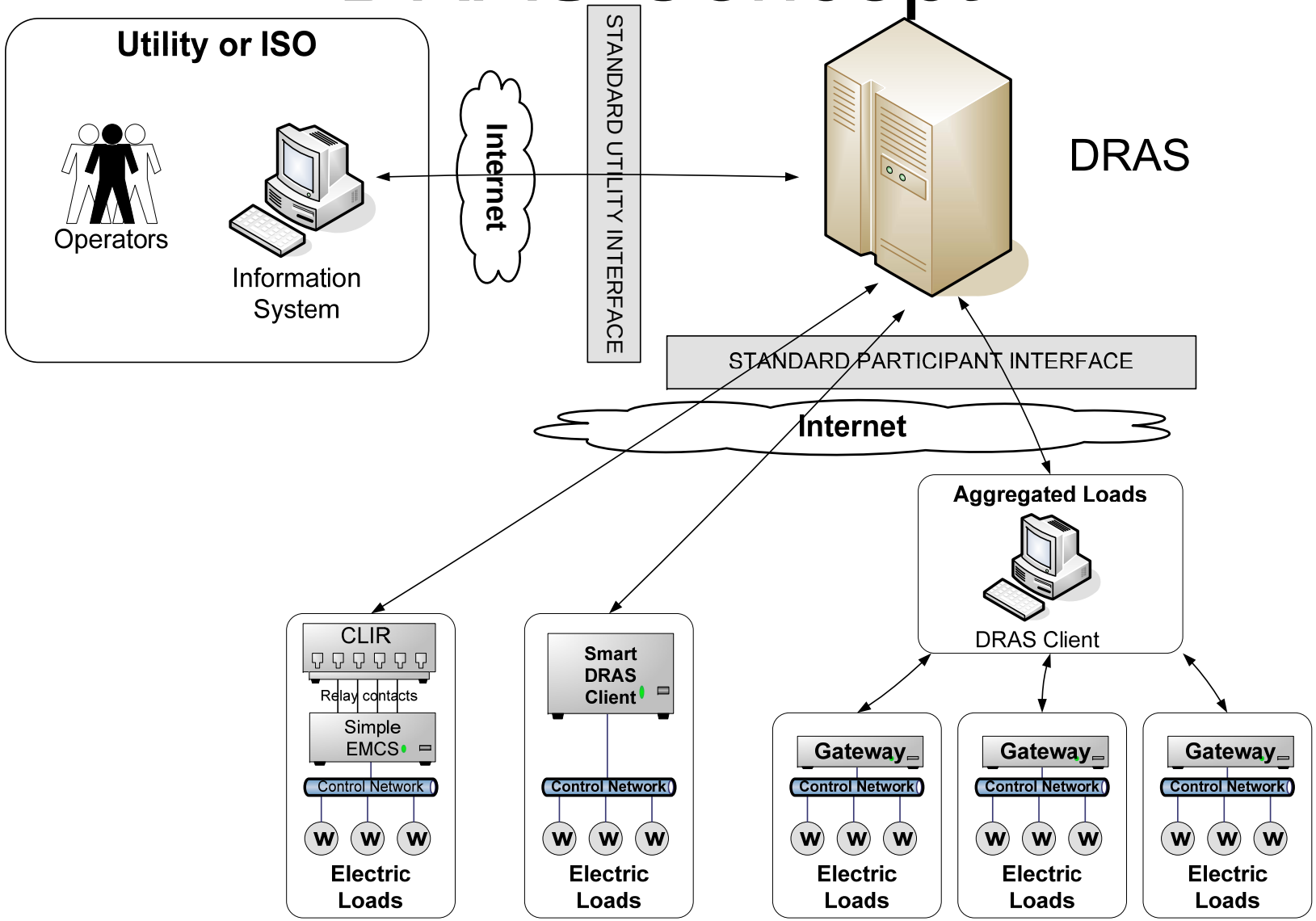
Tuesday, May 4, 3:30 -5:30

- Overview of existing 1.0 version of OpenADR specification and its data models, etc.

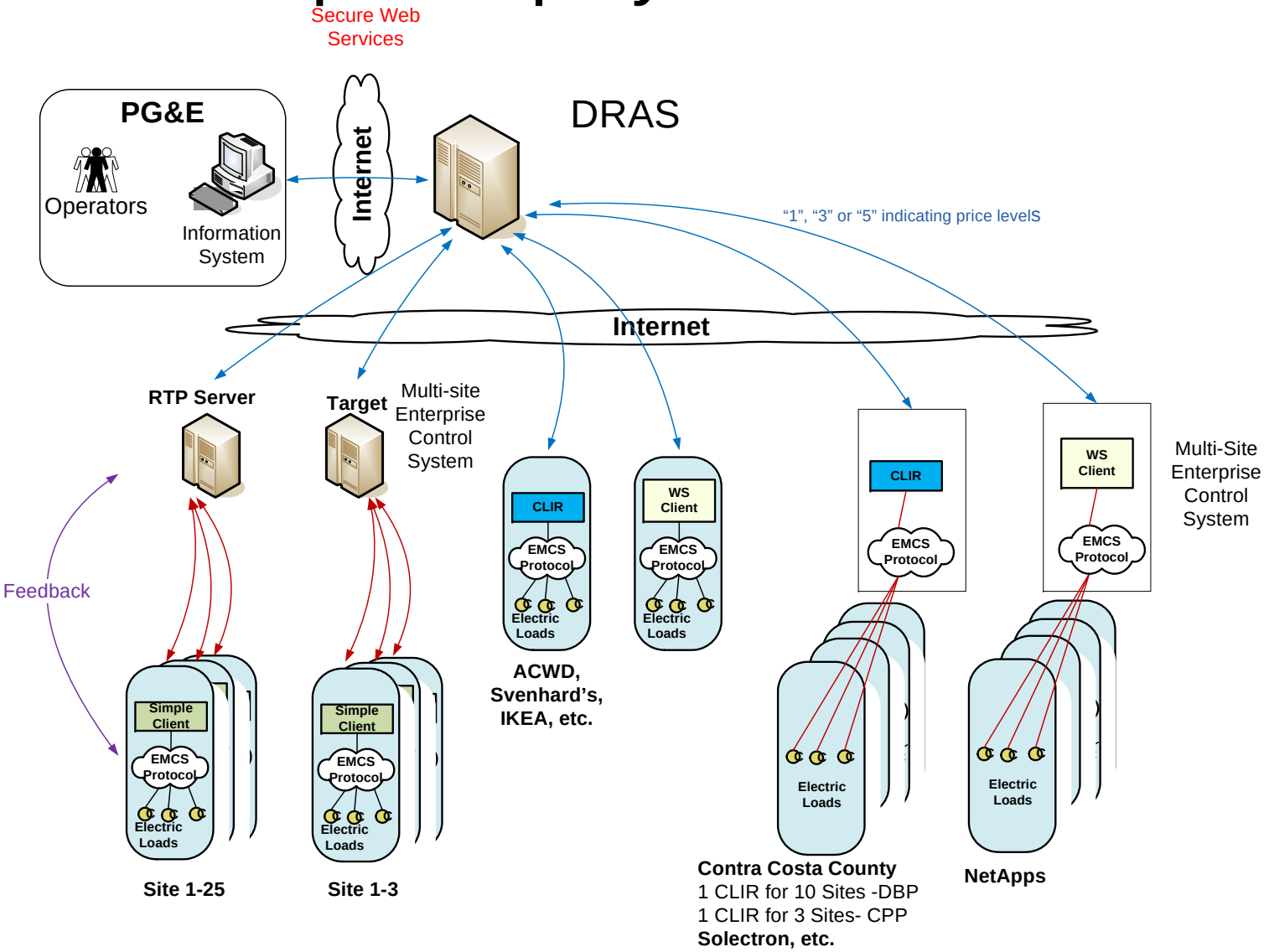
Definitions

- Utility/ISO – entity supplying electricity and offering DR programs
- Participant – Entity with business agreement with Utility/ISO to participate in DR programs
- DRAS – Demand Response Automation Server, automates delivery of messages between the Utility/ISO and the Participant.
- DRAS Client – entity owned by the Participant that communicates with the DRAS to receives DR Event information (MM communications).

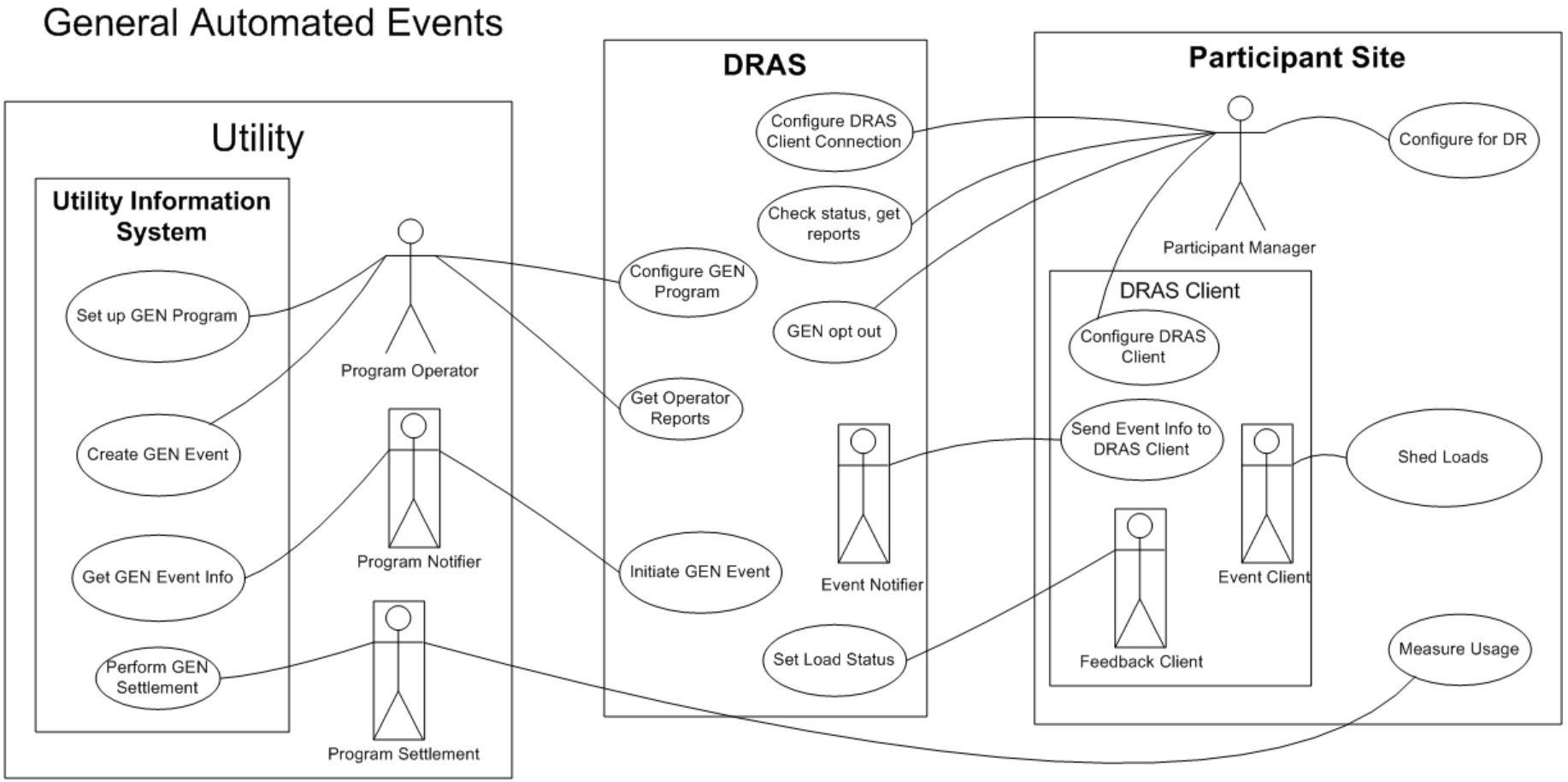
DRAS Concept



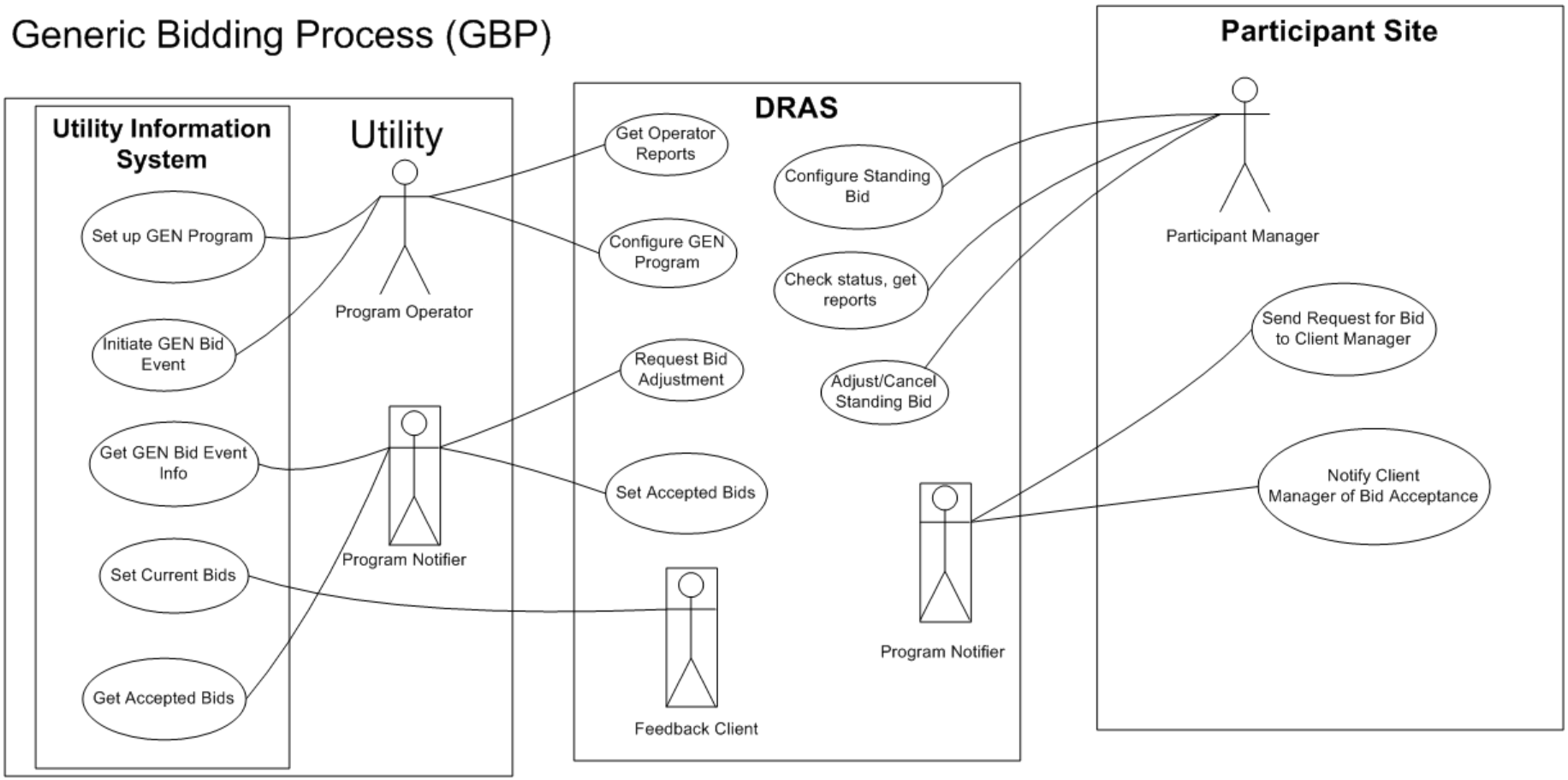
Sample Deployment Scenarios



Automated DR Events Uses Case



Automated Bidding Use Case



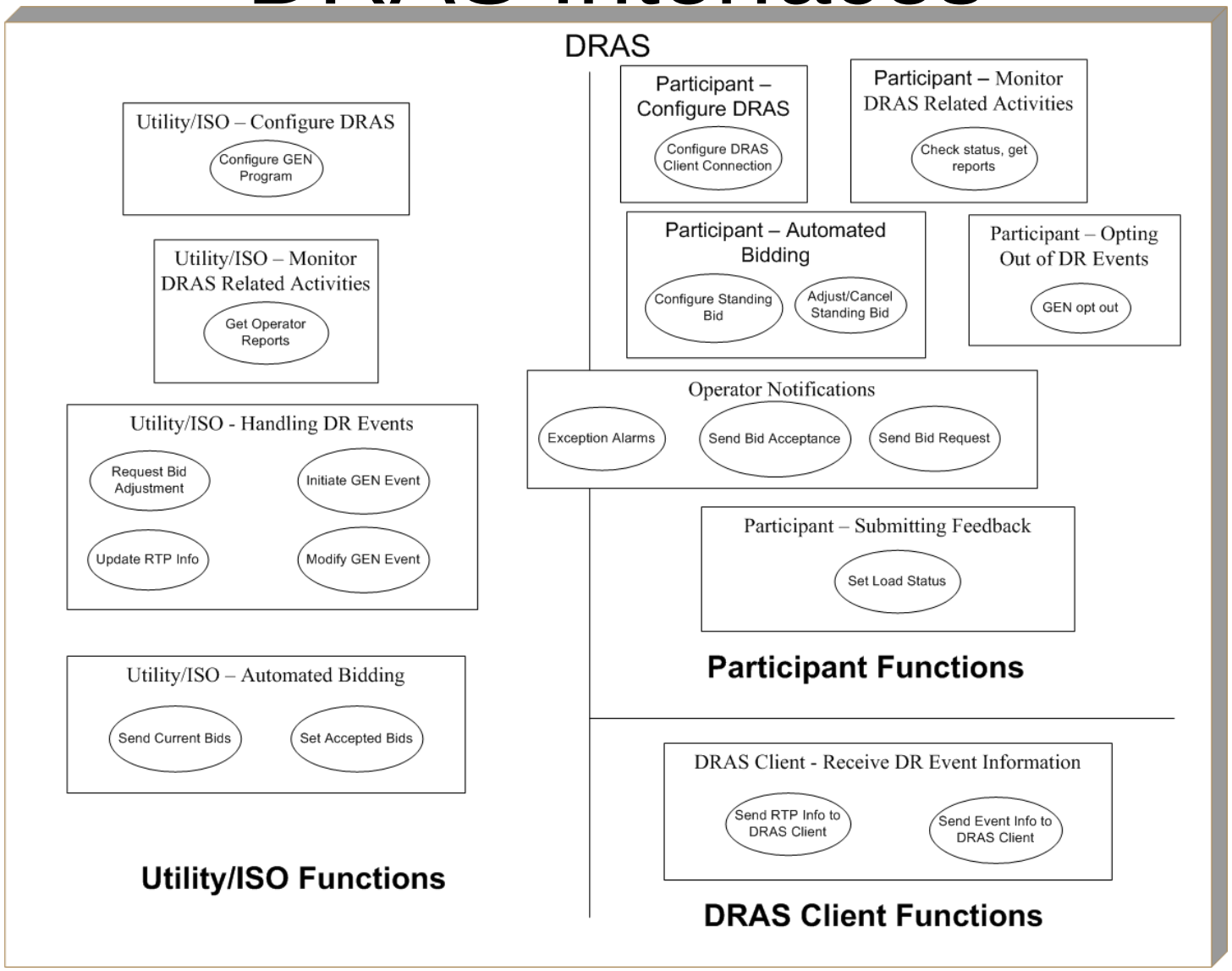
DRAS Requirements

- Communications with the DRAS should use readily available and existing networks such as the internet.
- The DRAS interfaces should be platform independent and leverage existing standards such as XML and Web Services.
- The DRAS communications should use a security policy that enables non-repudiation and encryption of the communications with the DRAS.
- The DRAS should support communications with a variety of control systems that may range from a very simple EMCS (Simple DRAS client) to those with sophisticated data processing and programming capabilities (Smart DRAS client).

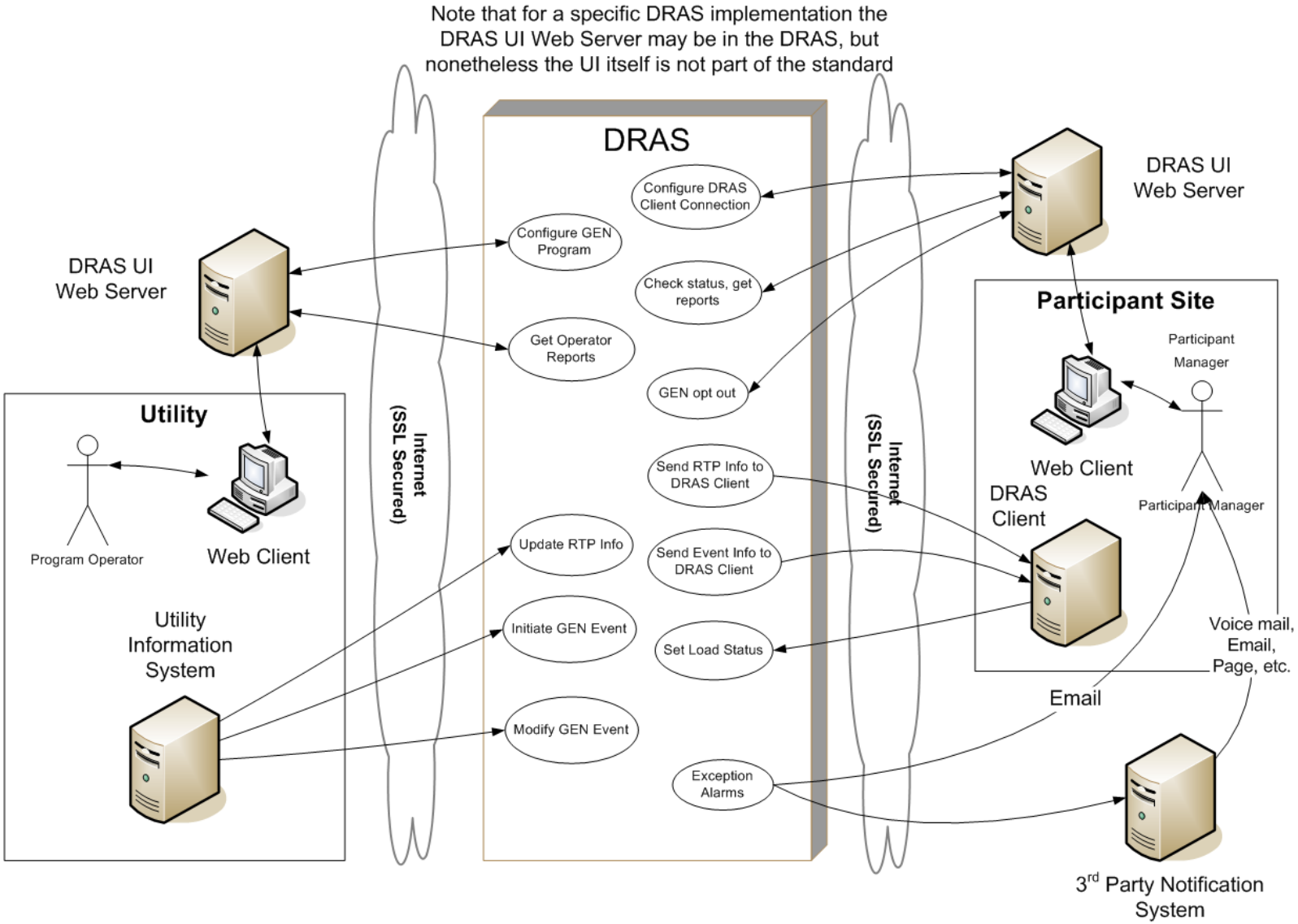
DRAS Requirements (cont)

- The DRAS should not be dependent on specific control systems within the facilities.
- DRAS Clients that communicate with the DRAS should easily integrate with existing facility networks and IT infrastructures.
- The DRAS should support aggregated loads that may be managed by third party aggregators.
- Reconciliation of DR Event participation is outside the scope of the DRAS. There are a number of methods such as aggregators, AMI, etc. that can and will handle the measurement of sheds for the purposes of the reconciliation of DR programs.

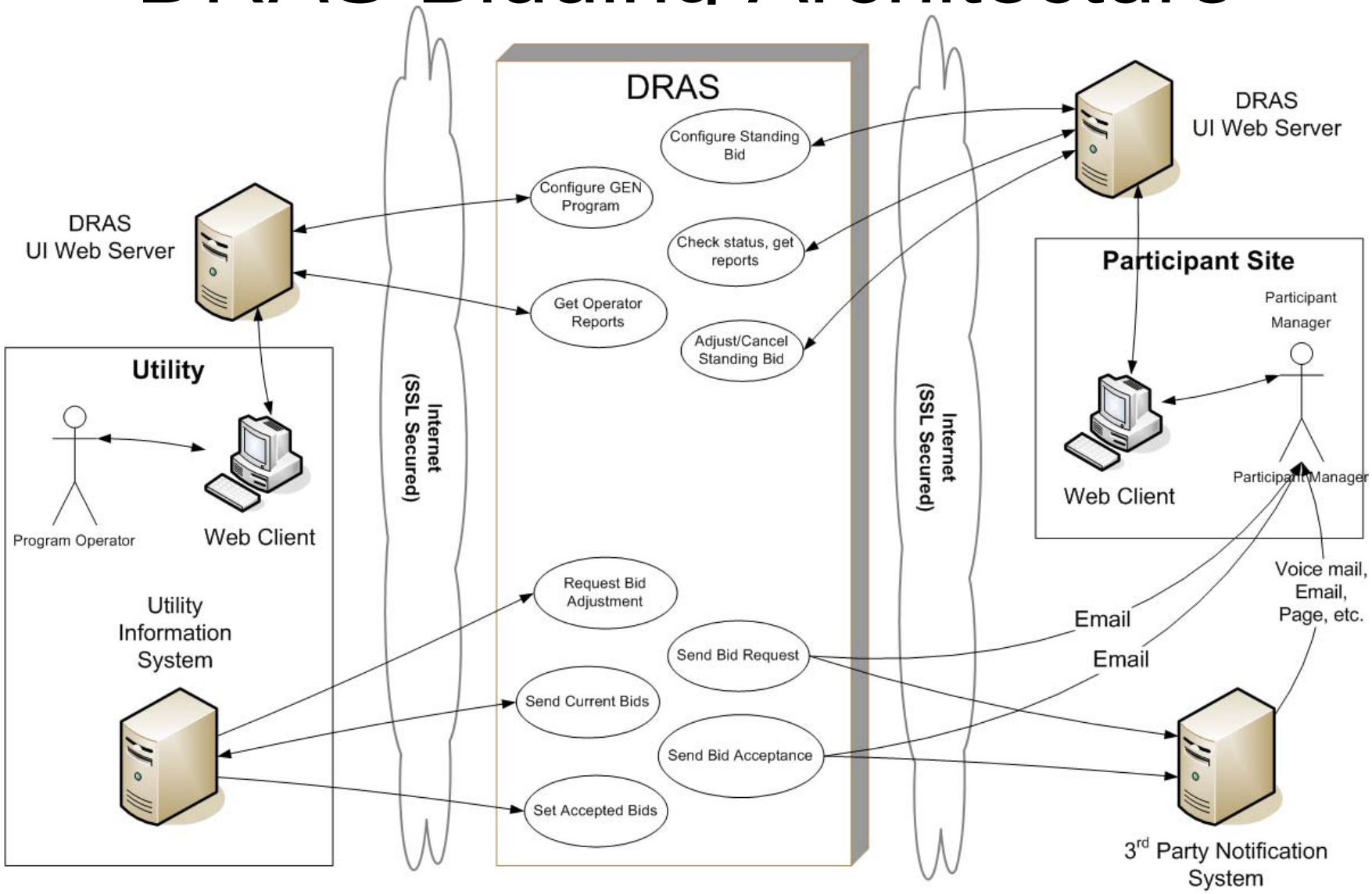
DRAS Interfaces



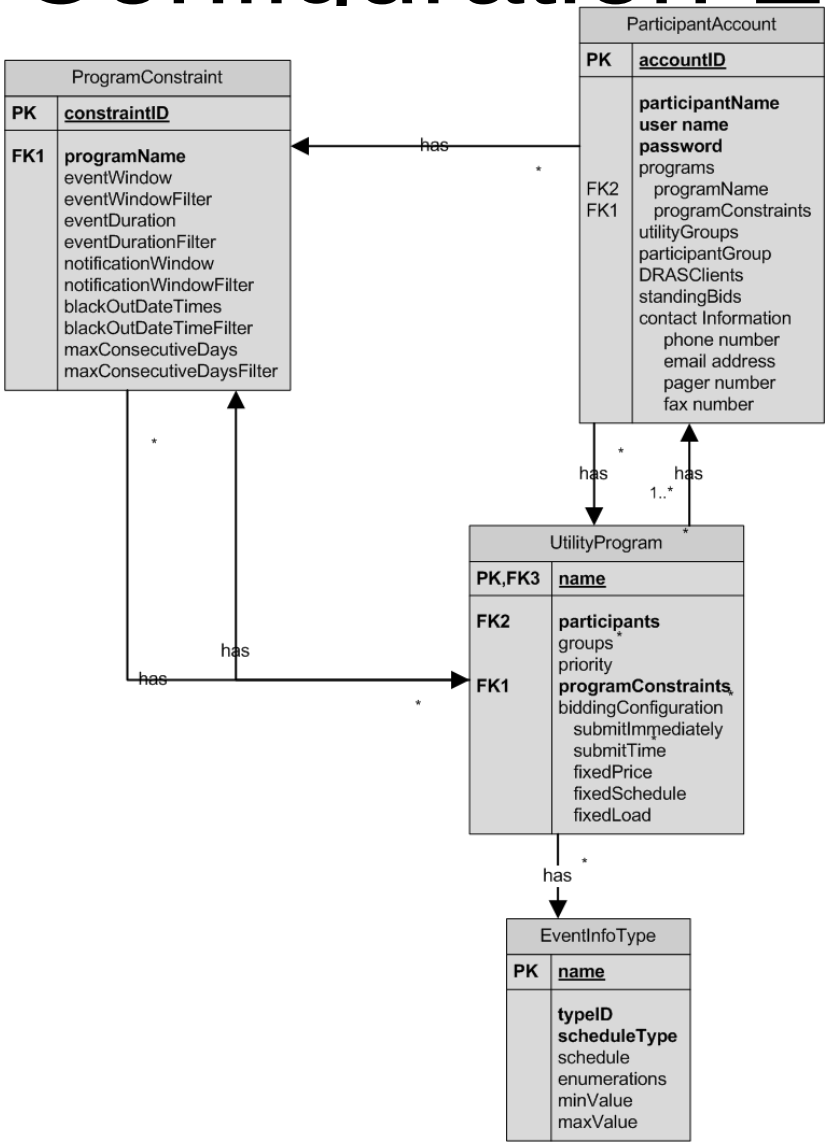
DRAS Event Architecture



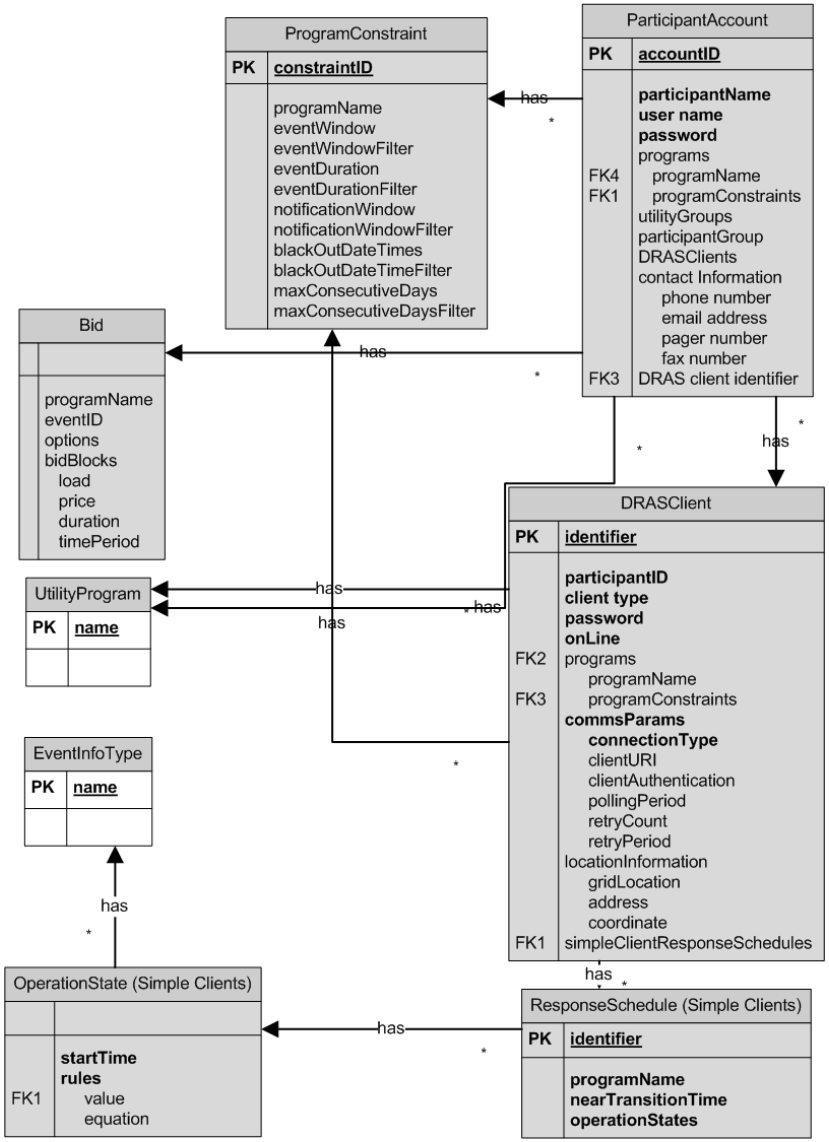
DRAS Bidding Architecture



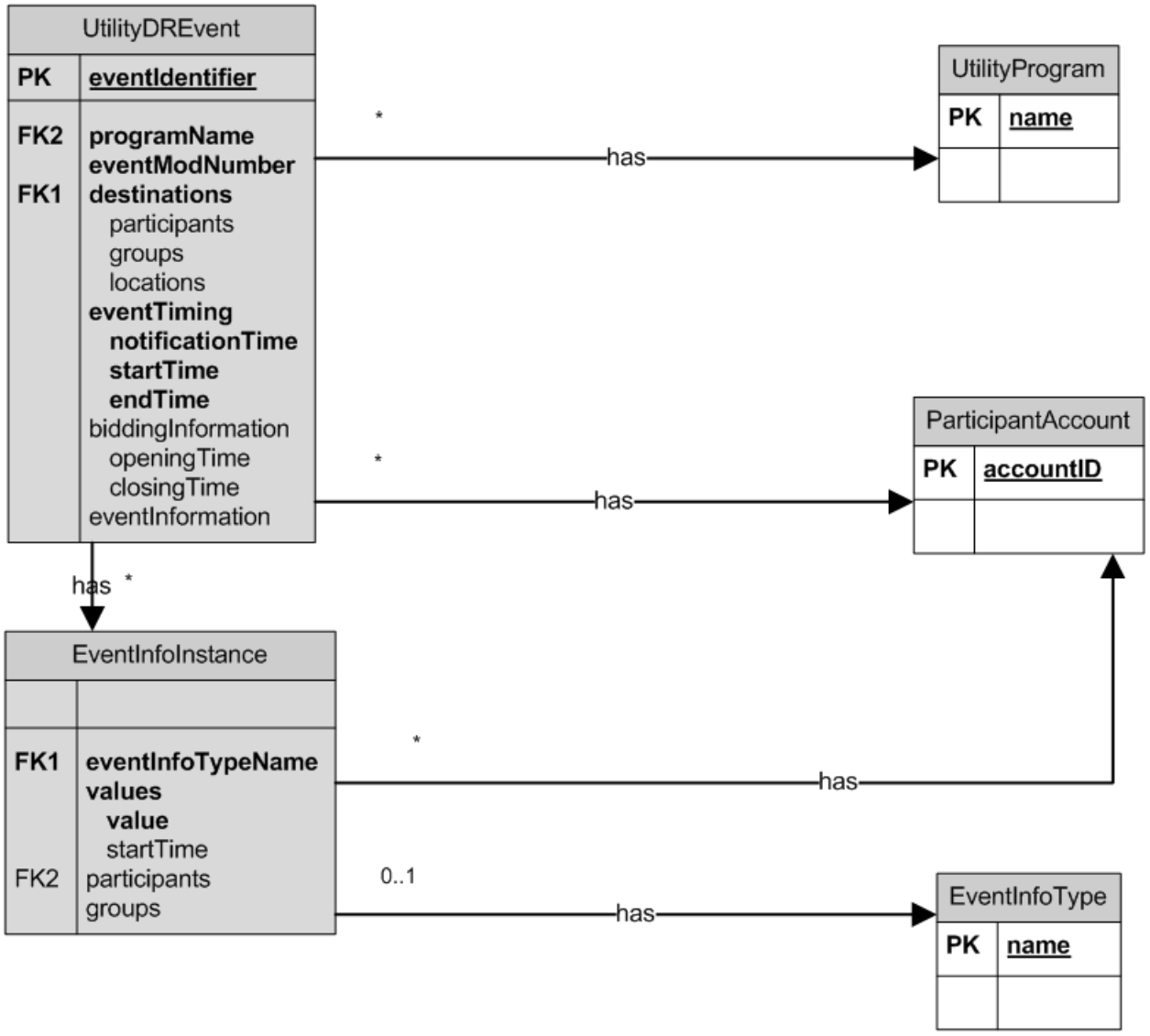
Utility Configuration Entities



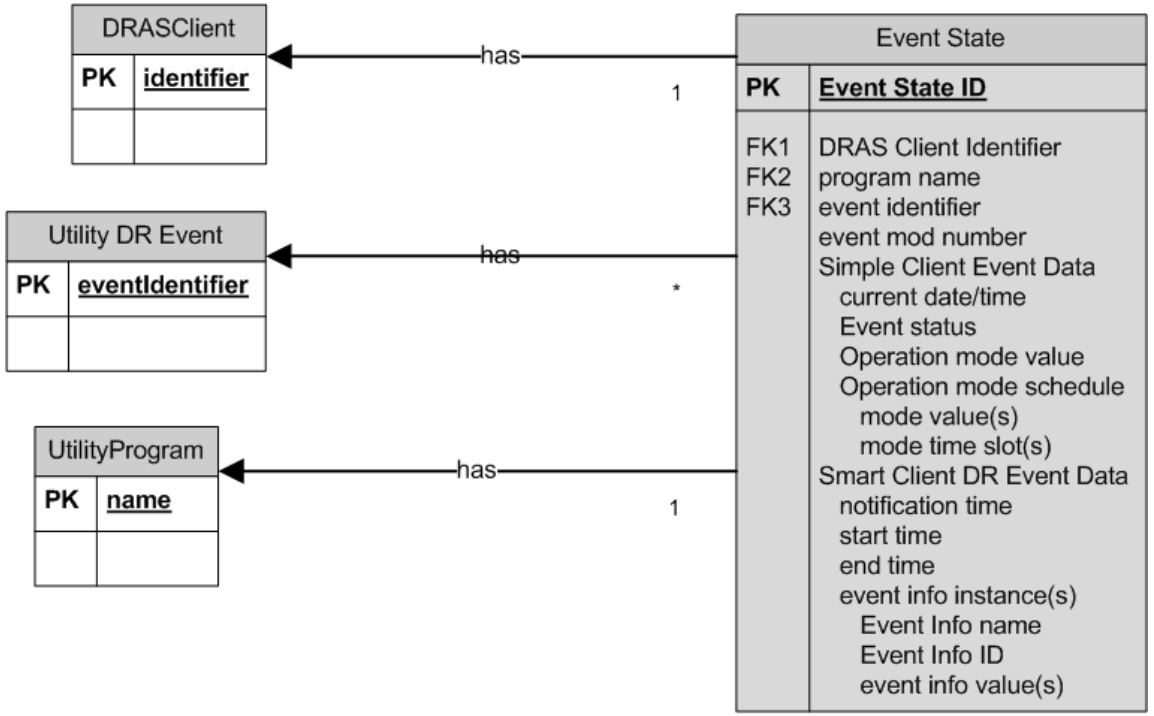
Participant Configuration Entities



Utility Issues DR Event

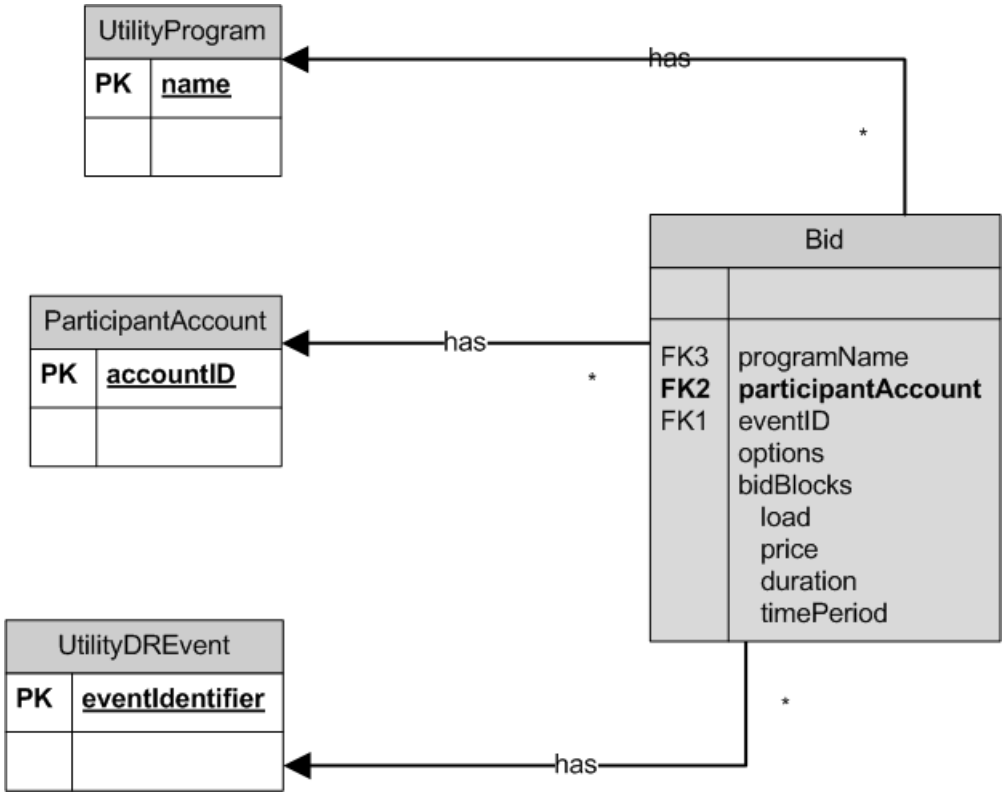


DRAS Client DR Event State

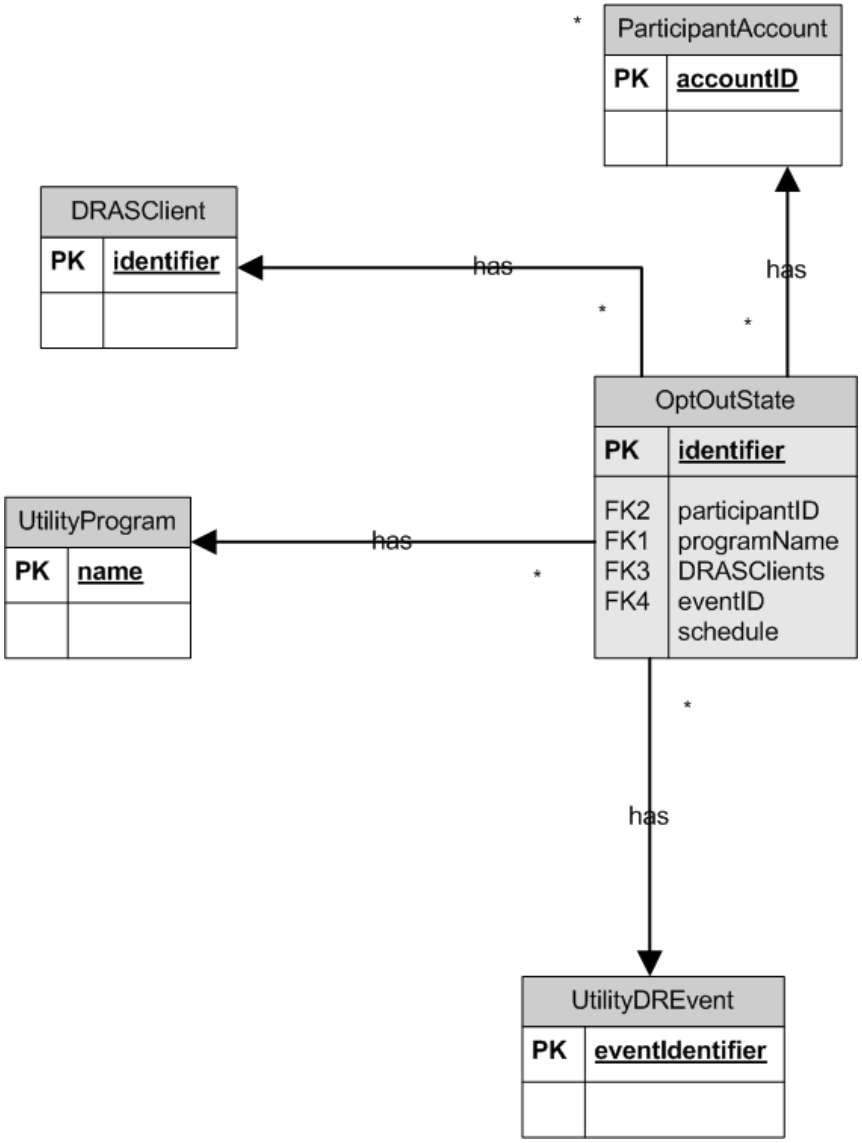


Event State Confirmation	
PK	**Event State ID**
	DRAS Client Identifier program name event identifier event mod number current date/time operation mode value opt-in

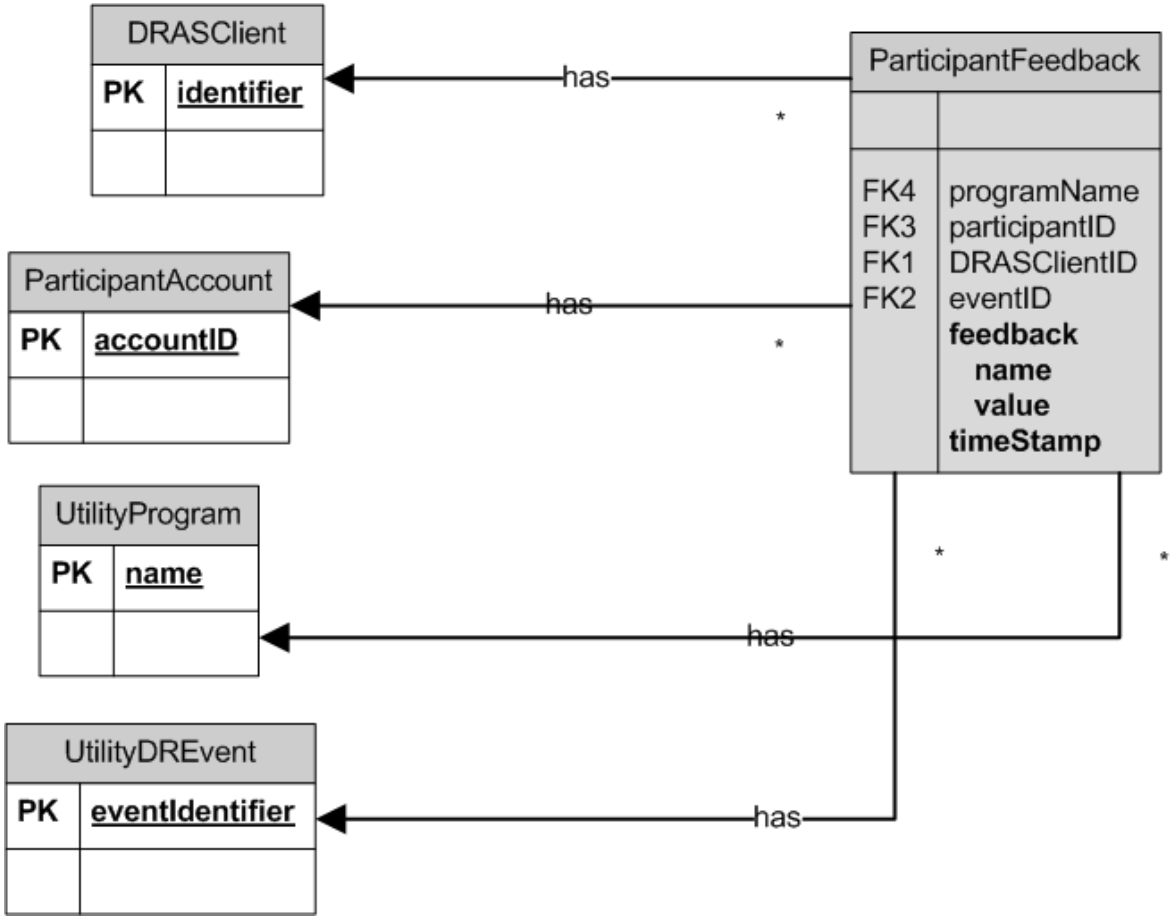
Participant Submits Bid



Participant Opt-Out

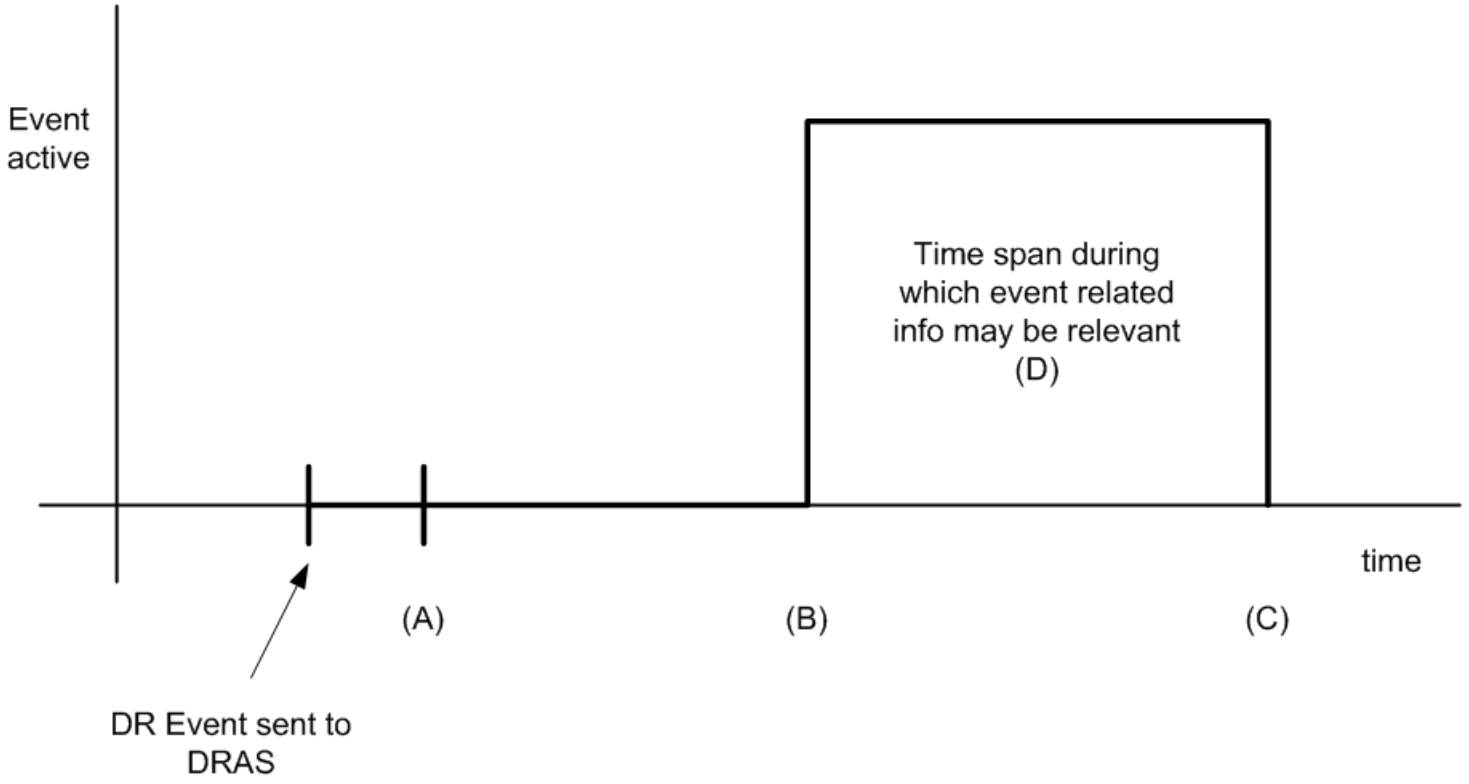


Participant Feedback

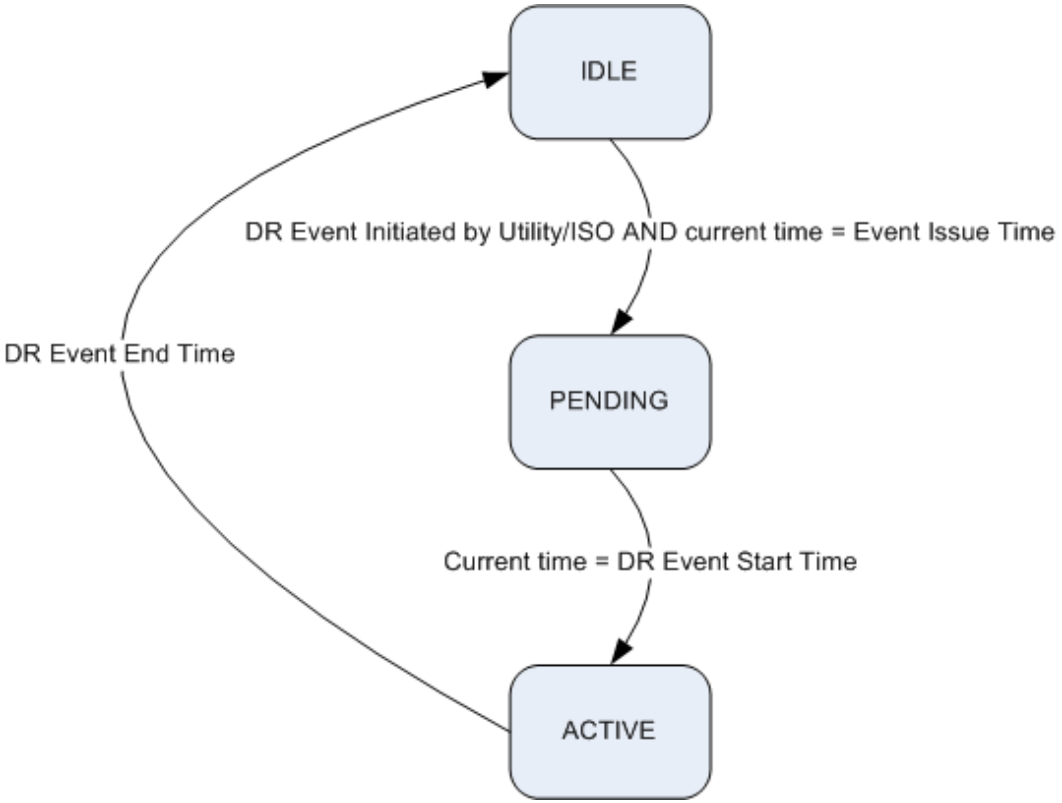


DR Event Model

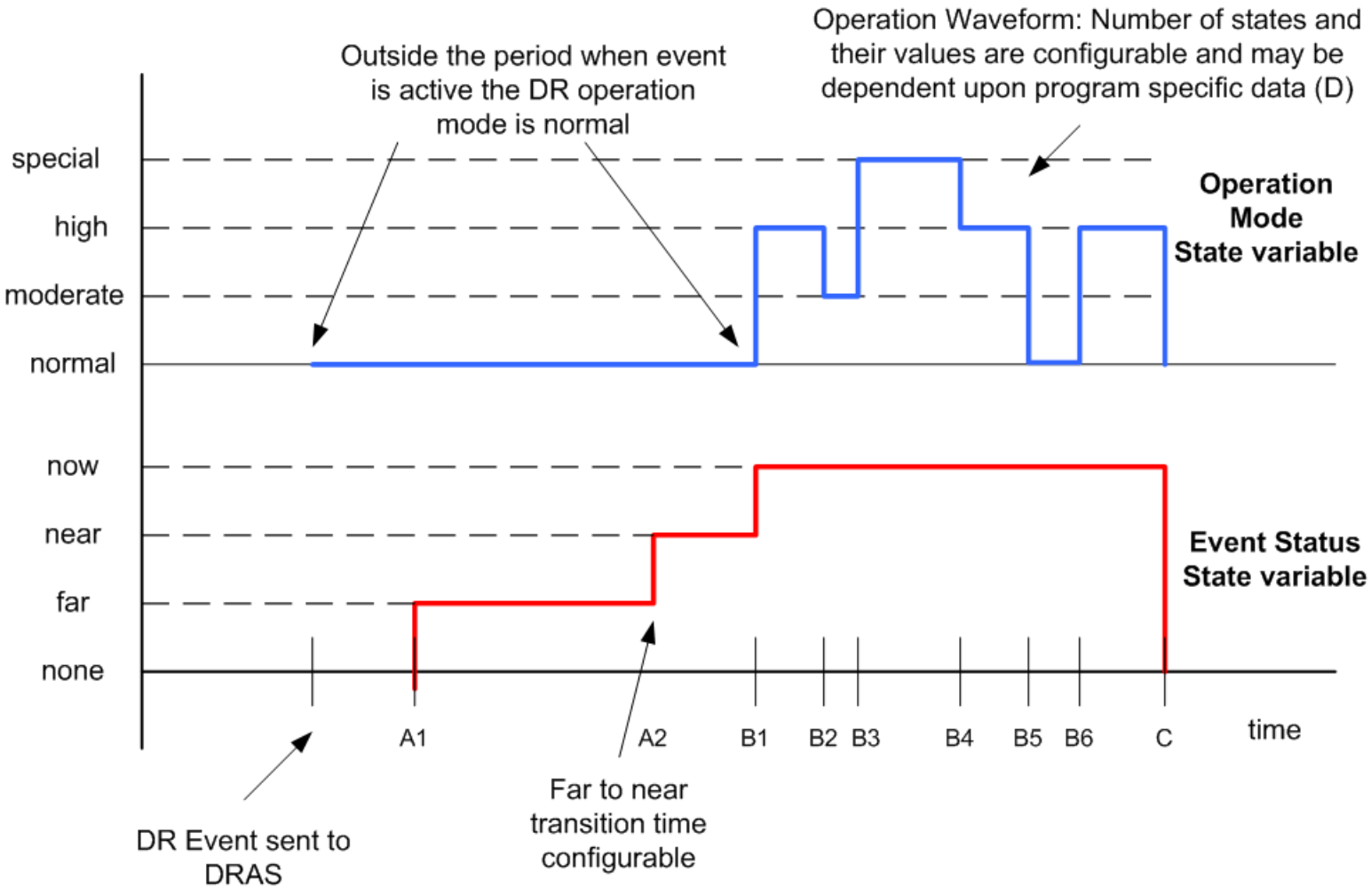
(A) Issue Time – Time at which participants should be notified of an upcoming event.
(B) Start Time – Time at which the event starts.
(C) End Time – Time at which the event ends.
(D) Event Info – Program specific information that is related to the event, e.g. RTPor level.
name value pair, start/end time, (constrained within event period). Note that the Event Info may also be used to set up conditional events.



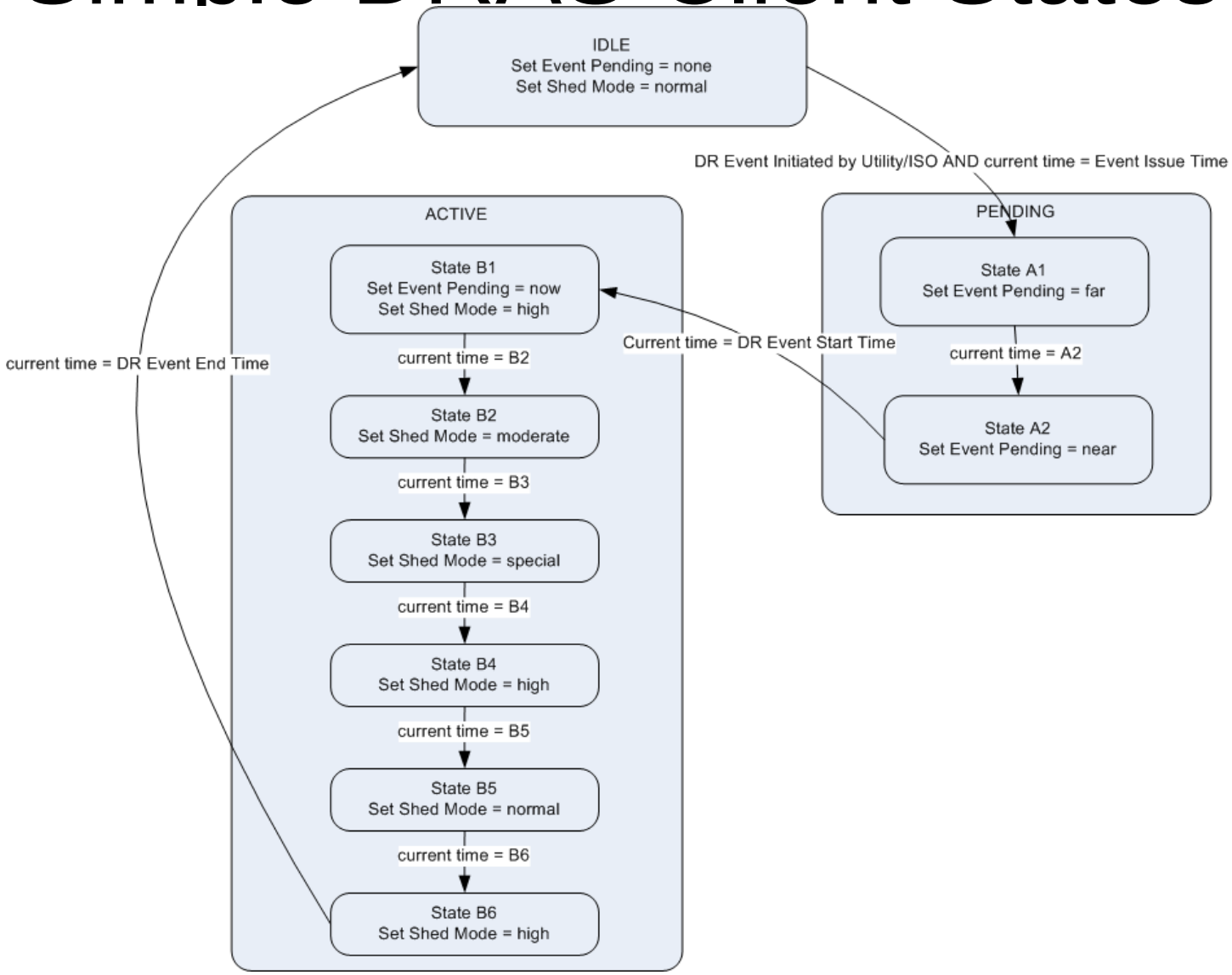
Event States



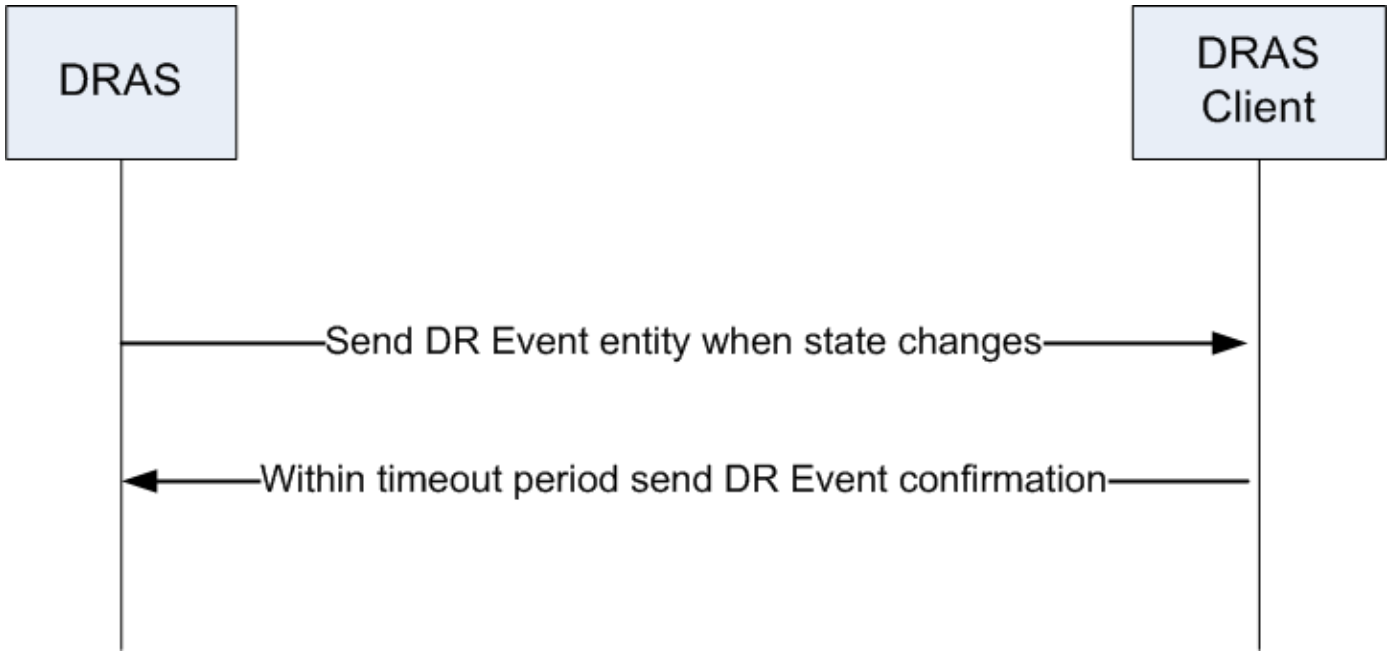
Simple DRAS Client Model



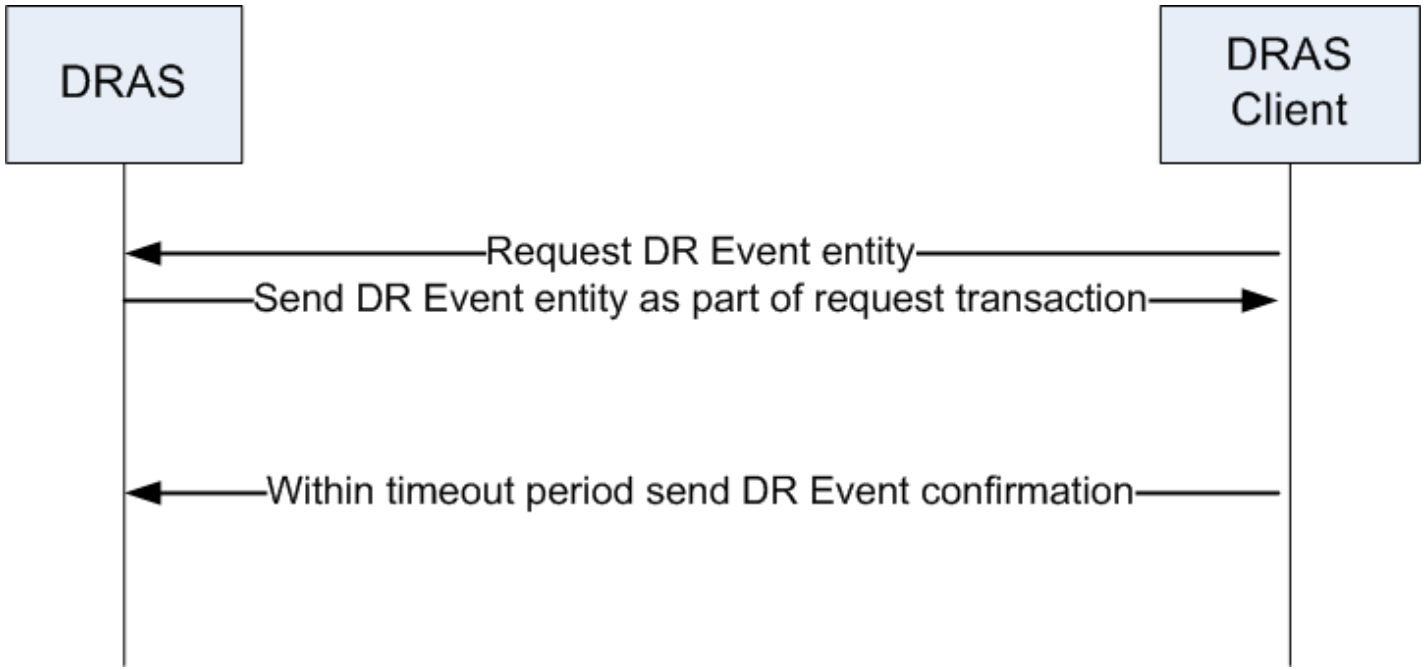
Simple DRAS Client States



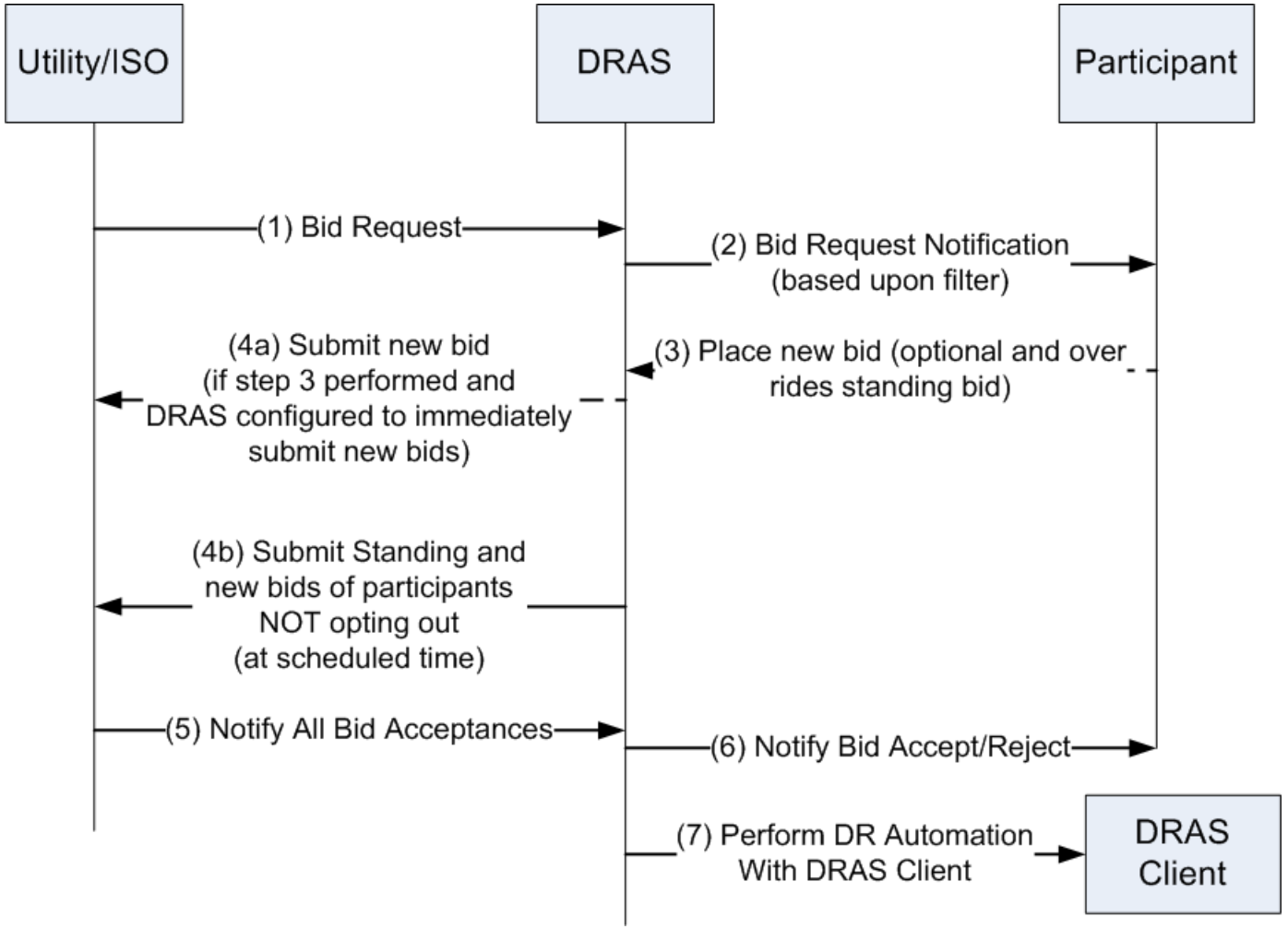
DRAS Client Interaction (PUSH)



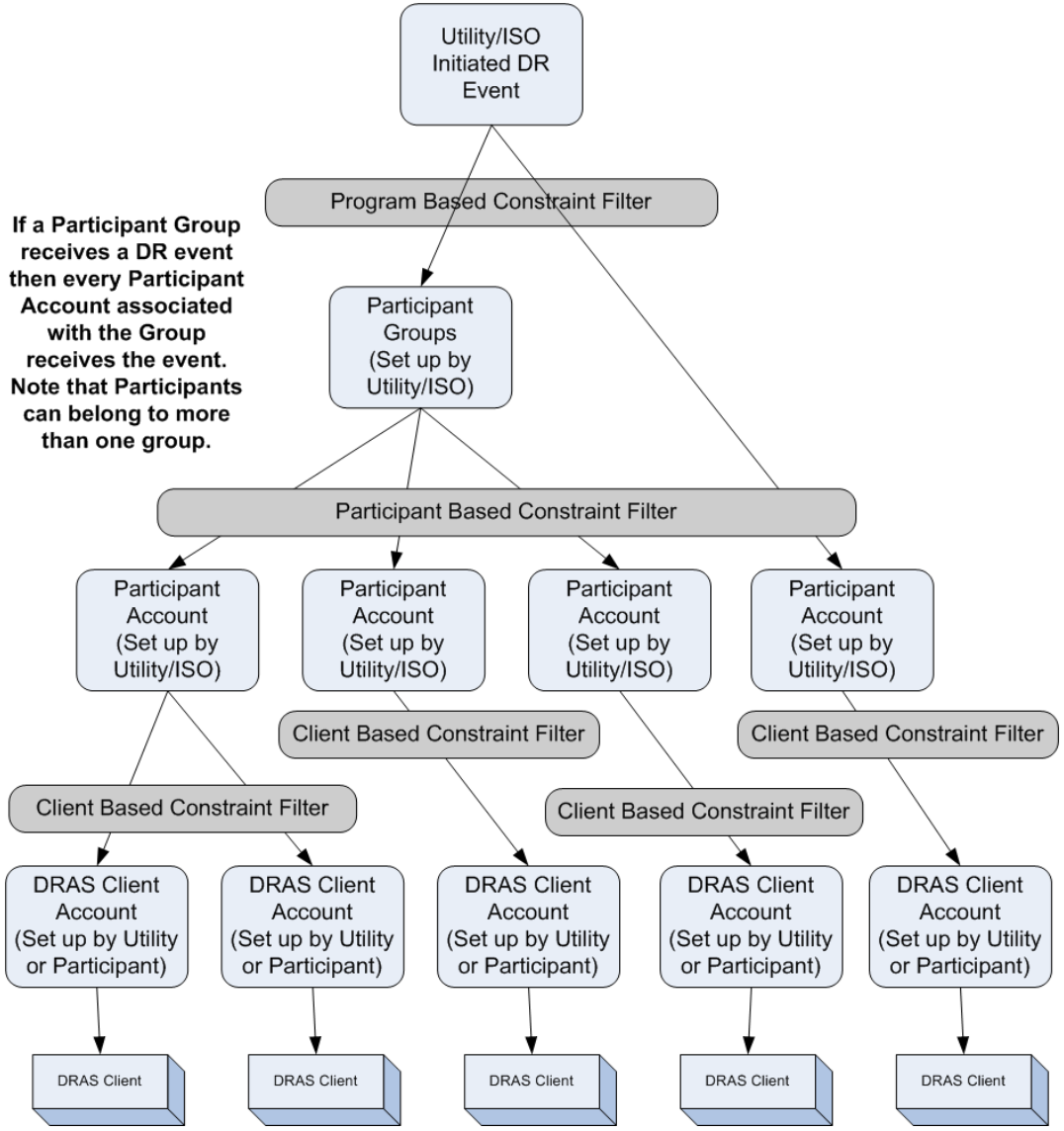
DRAS Client Interaction (PULL)



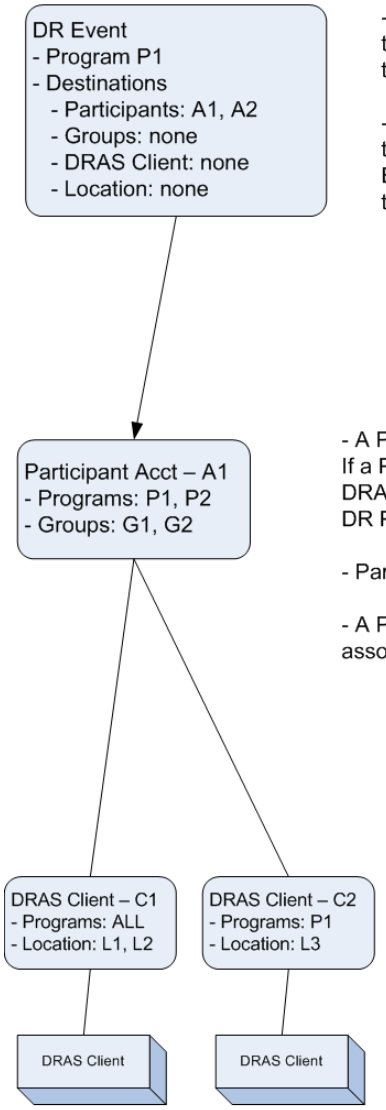
Bidding Sequence Diagram



Event Propagation Model



Event Propagation



- DR Events are associated with a particular Program. Only those DRAS Clients that belong to that Program will receive that event.

- DR Events have a “destination” specification that is used by the DRAS to determine which DRAS Clients receive the DR Event. The destination specification may have one or more of the following attributes:
    - List of participants.
    - List of Utility groups.
    - List of DRAS Clients.
    - List of locations.

- A Participant Account may belong to one or more DR Programs. If a Participant Account does not belong to a DR Program then the DRAS Clients associated with it will not receive DR Events for that DR Program.

- Participant Accounts may belong to one or more Groups.

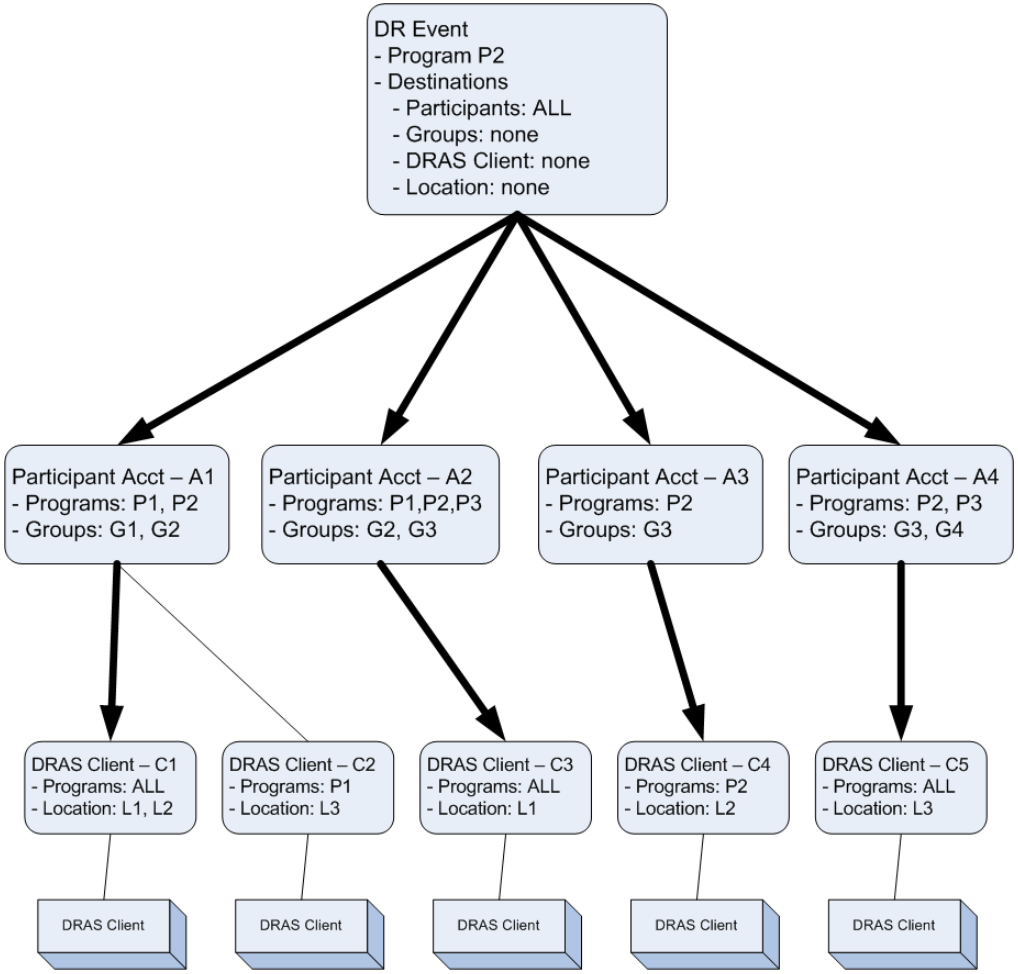
- A Participant Account may have one or more than DRAS Clients associated with it.

- DRAS Clients are associated with one and only one Participant.

- By default a DRAS Client belongs to all the same Programs that the Participant Account that it is associated with belongs to, but it may also belong to a subset of such Programs. If a DRAS Client does not belong to a DR Program then it will not receive DR Events for that DR Program.

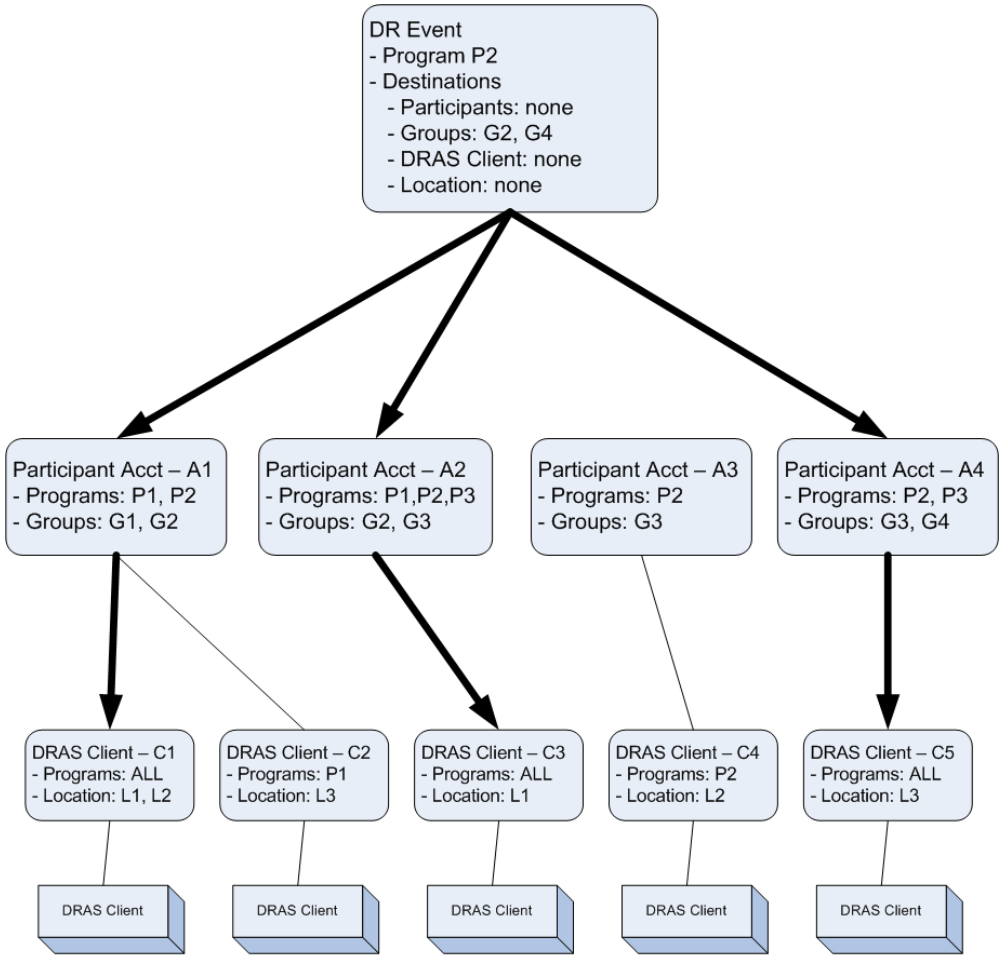
- A DRAS Client may have a location specification consisting of one or more of the following attributes:
    - grid location
    - address
    - lat/lon coordinates

Event Propagation Example



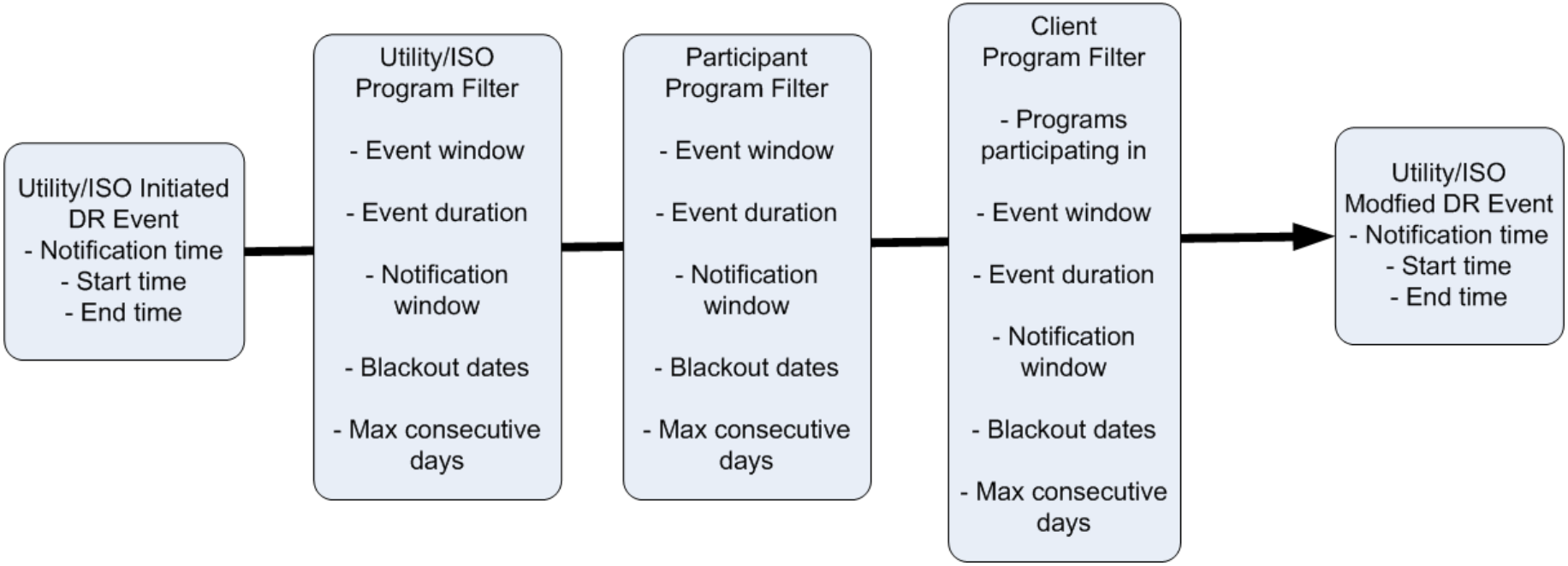
Example: DR Event for Program P2 - All Participant Accounts

Example Event Propagation

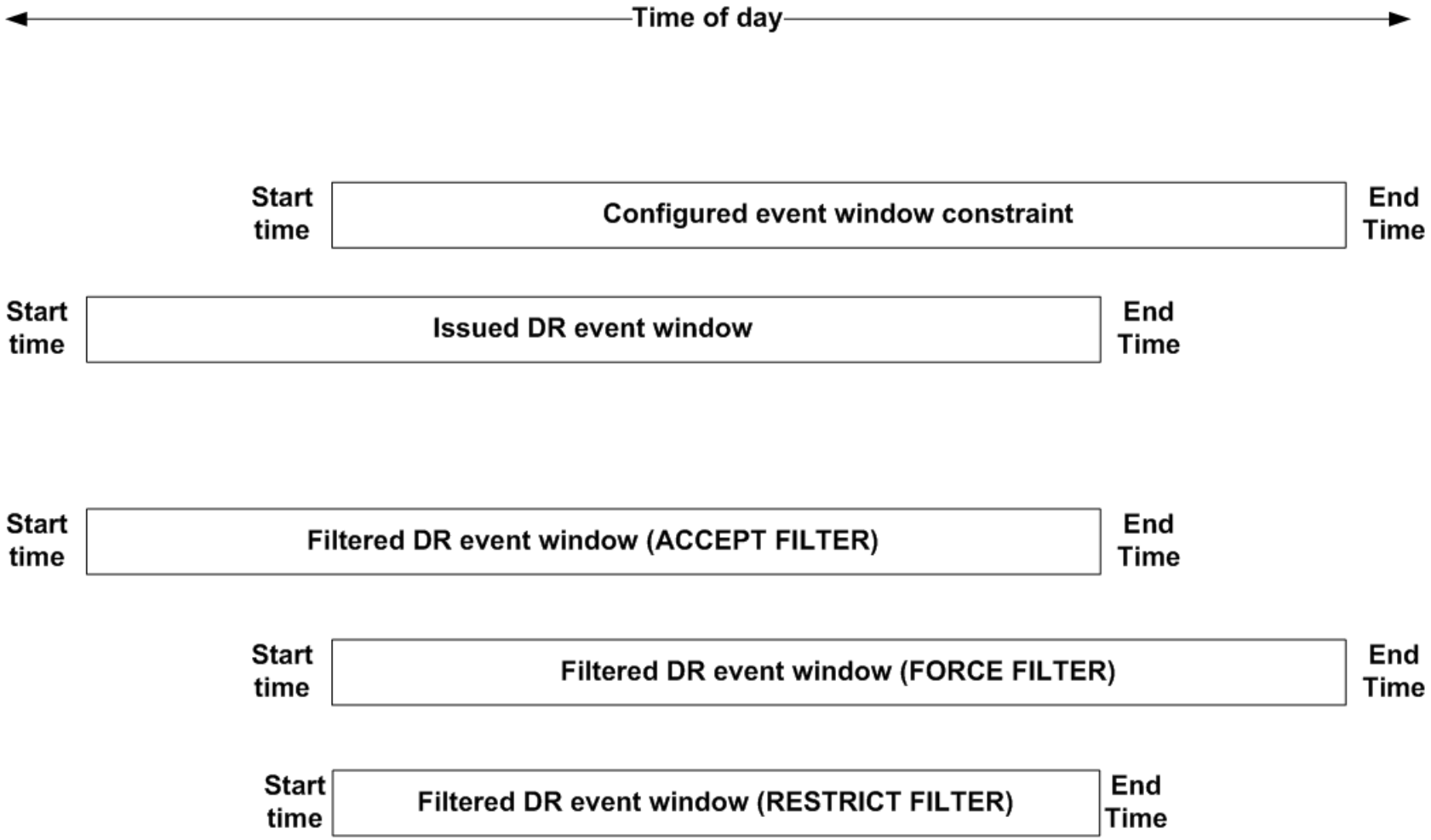


Example: DR Event for Program P2 –
Groups G2, G4

Program Constraints



Event Window Constraint



API Interfaces

- Utility/ISO Interface
- Participant Interface
- DRAS Client Interface

Utility/ISO Methods for Handling DR Events

- InitiateDREvent
- ModifyDREvent
- AdjustDREventParticipants
- GetDREventInformation
- SetEventConstraint
- GetEventConstraint

Utility/ISO Methods to Support Automated Bidding

- GetCurrentBids (PULL MODEL)
- SetCurrentBids (PUSH MODEL)
- CloseBidding
- SetBidStatus

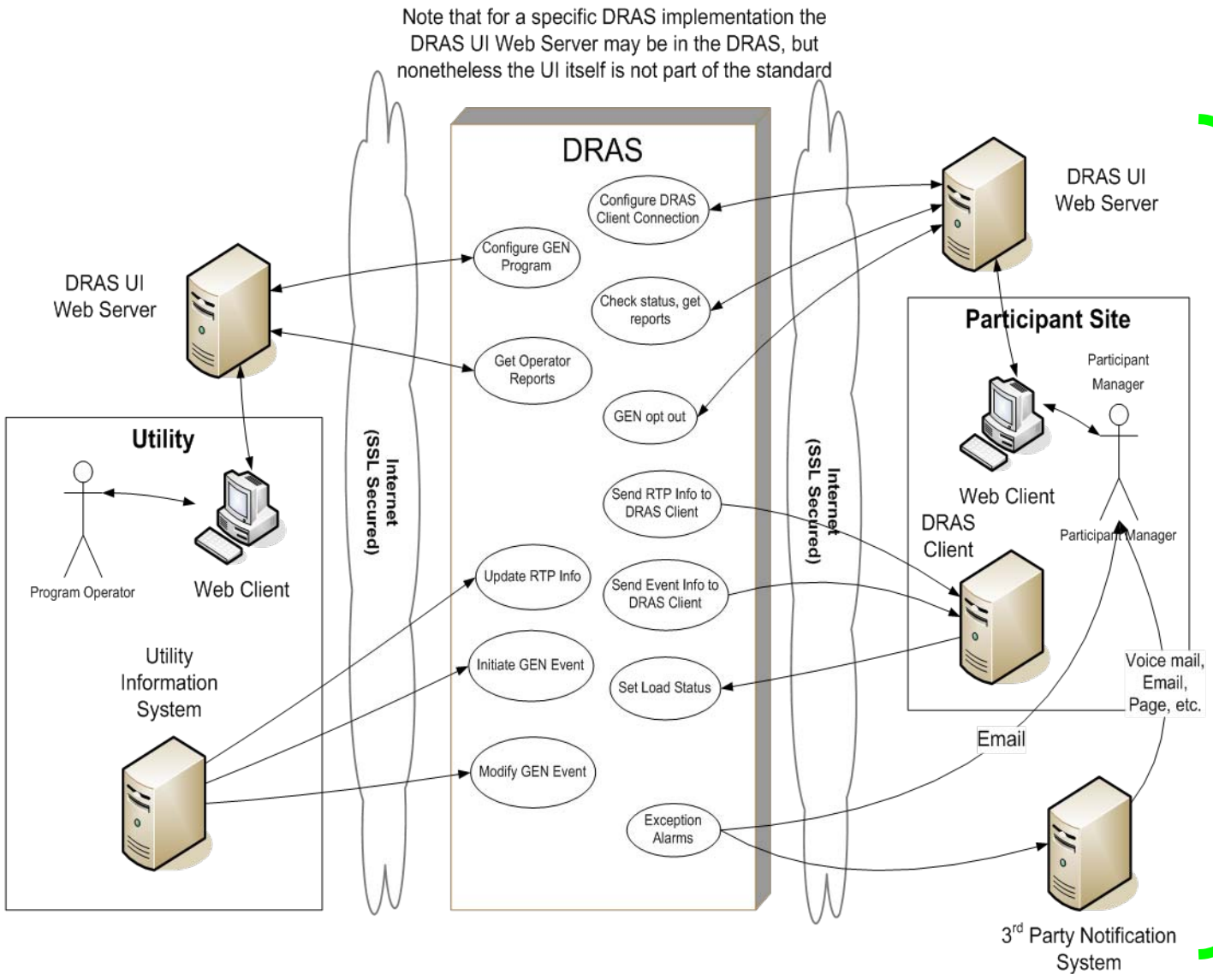
Utility/ISO Methods to Configure DRAS

- Manage Programs
  - CreateProgram
  - ModifyProgram
  - DeleteProgram
  - GetPrograms
  - AdjustProgramParticipants
- Manage Participant Accounts
  - CreateParticipantAccounts
  - ModifyParticipantAccounts
  - DeleteParticipantAccounts
  - GetParticipantAccounts
  - GetGroups

Utility/ISO Monitoring of DRAS Related Activities

- GetDRASClientCommsStatus
- GetDRASTransactions
- GetDRASClientAlarms
- GetParticipantFeedback

Participant, Client, and DRAS UI



Methods

Relationships

Actors

Roles

Use Cases

Data Models

> SOAP (2-way) & REST (PULL only) Support

Participant Op Interface…

DRAS

Participant – Configure DRAS

Configure DRAS Client Connection

Participant – Monitor DRAS Related Activities

Check status, get reports

Participant – Automated Bidding

Configure Standing Bid

Adjust/Cancel Standing Bid

Participant – Opting Out of DR Events

GEN opt out

Operator Notifications

Exception Alarms

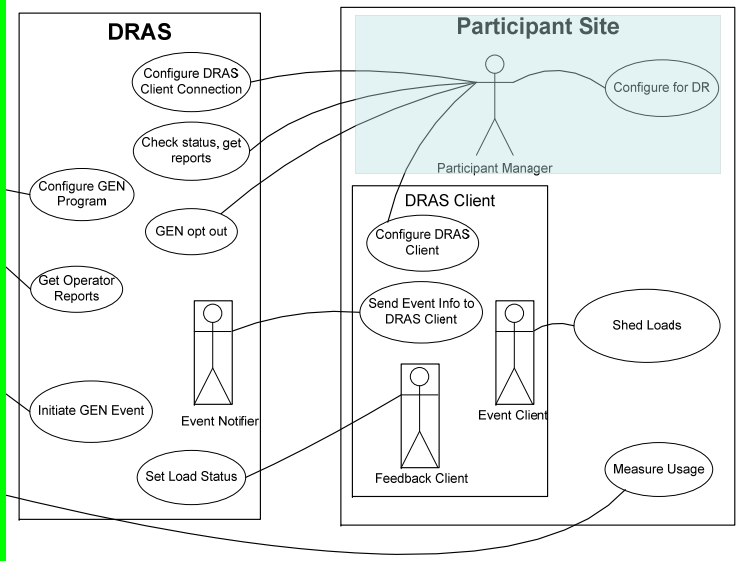
Send Bid Acceptance

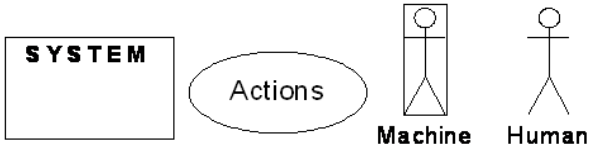
Send Bid Request

Participant – Submitting Feedback

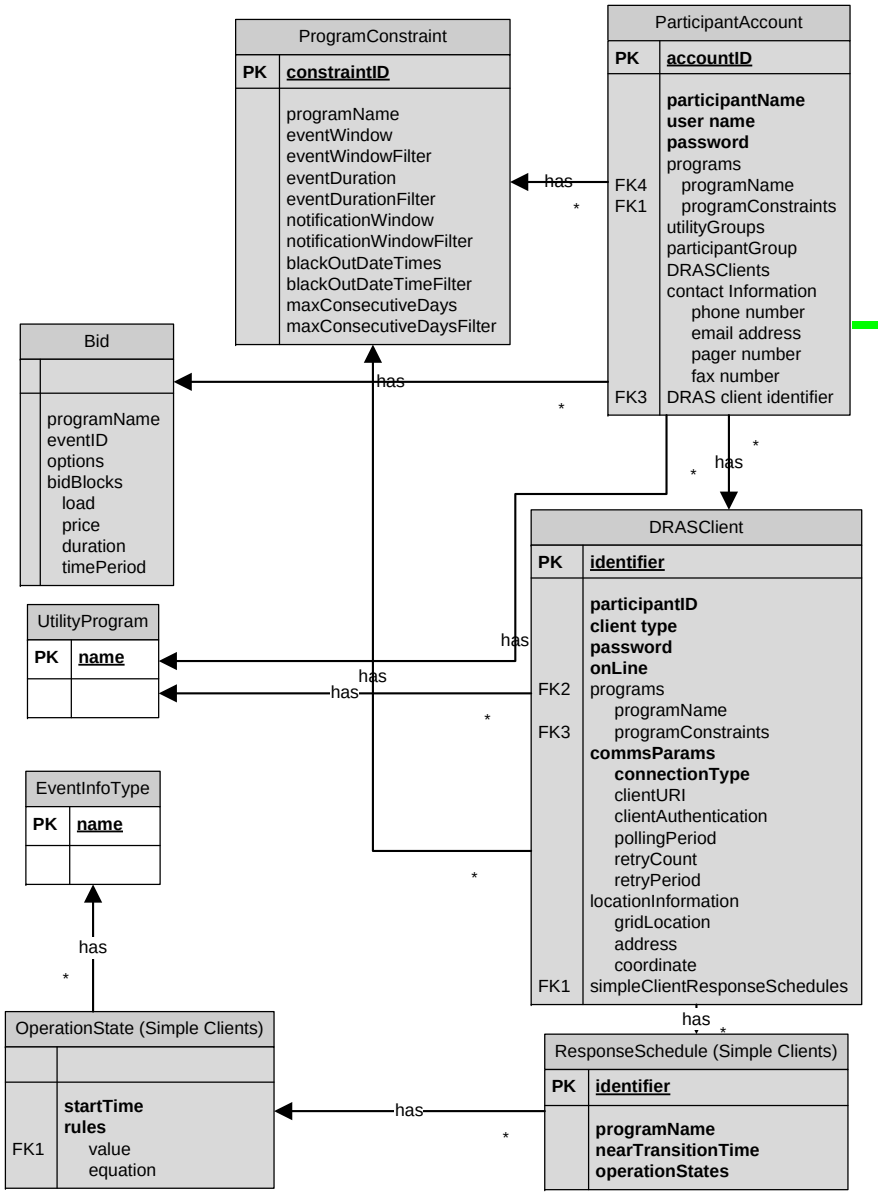
Set Load Status

Participant Functions



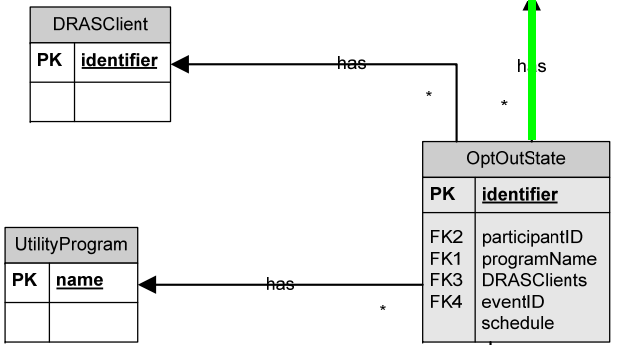


Participant Op Interface



Participant Configuration Entities

- ***ParticipantAccount*** entity created by Utility/ISO
- Participant may define ***ProgramConstraints***
- A *ParticipantAccount* may have standing ***Bid*** entities
- A *ParticipantAccount* may define ***DRASClient*** entities
- **Participant Operator** configures:
  - ***[ParticipantFeedback*** entity]
  - **Logs** & Exceptions/**Alarms**



Methods & Their Functions

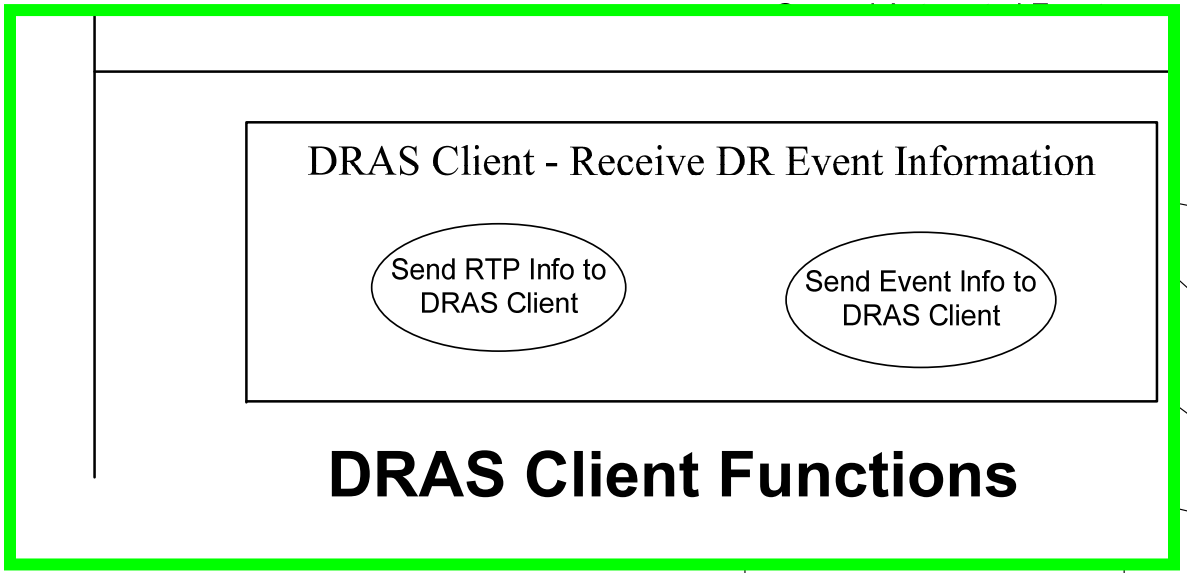
Participant Op Interface

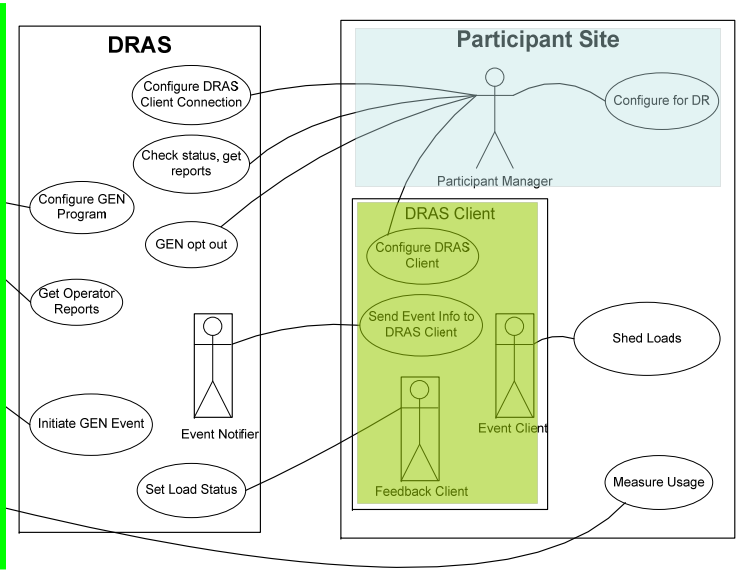
- **Configure Participant Info in DRAS**
  - **Manage Participant Account:** *GetParticipantAccounts, ModifyParticipantAccount*
  - **Manage DRAS Client:** *CreateDRASClient, ModifyDRASClient, DeleteDRASClient, GetDRASClientInfo*
  - **Manage Program Constraint:** *GetParticipantProgramConstraints, SetParticipantProgramConstraints, DeleteParticipantProgramConstraints, GetDRASClientProgramConstraints, SetDRASClientProgramConstraints, DeleteDRASClientProgramConstraints*
  - **Manage Simple Client [Response Schedules]:** *GetProgramInformation, CreateResponseSchedule, DeleteResponseSchedule, GetResponseSchedule*
- **Opt-Out of DR Events:** *OptOutState* Entity
  - *CreateOptOutState, DeleteOptOutState, GetOptOutState*
- **Feedback (Facility Status) to DRAS**
  - *SetDREventFeedback, GetDREventFeedback*
- **Automated Bidding:** *Bid* Entity
  - *SubmitStandingBid, GetStandingBid,DeleteStandingBid, SubmitBid, GetBid*
- **Monitor DRAS-related Activities**
  - *GetDRASClientCommsStatus, GetDRASTransactions, GetDRASClientAlarms*
- **Install & Test DRAS Clients**
  - *SetTestMode, SetTestModeState, GetTestModeState*

OpenADR Standards

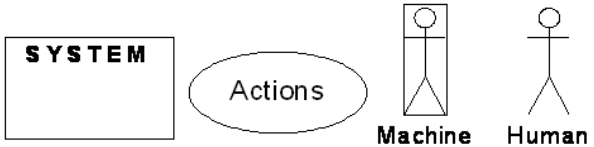
DRAS Client Interface

DRAS Client Interface…

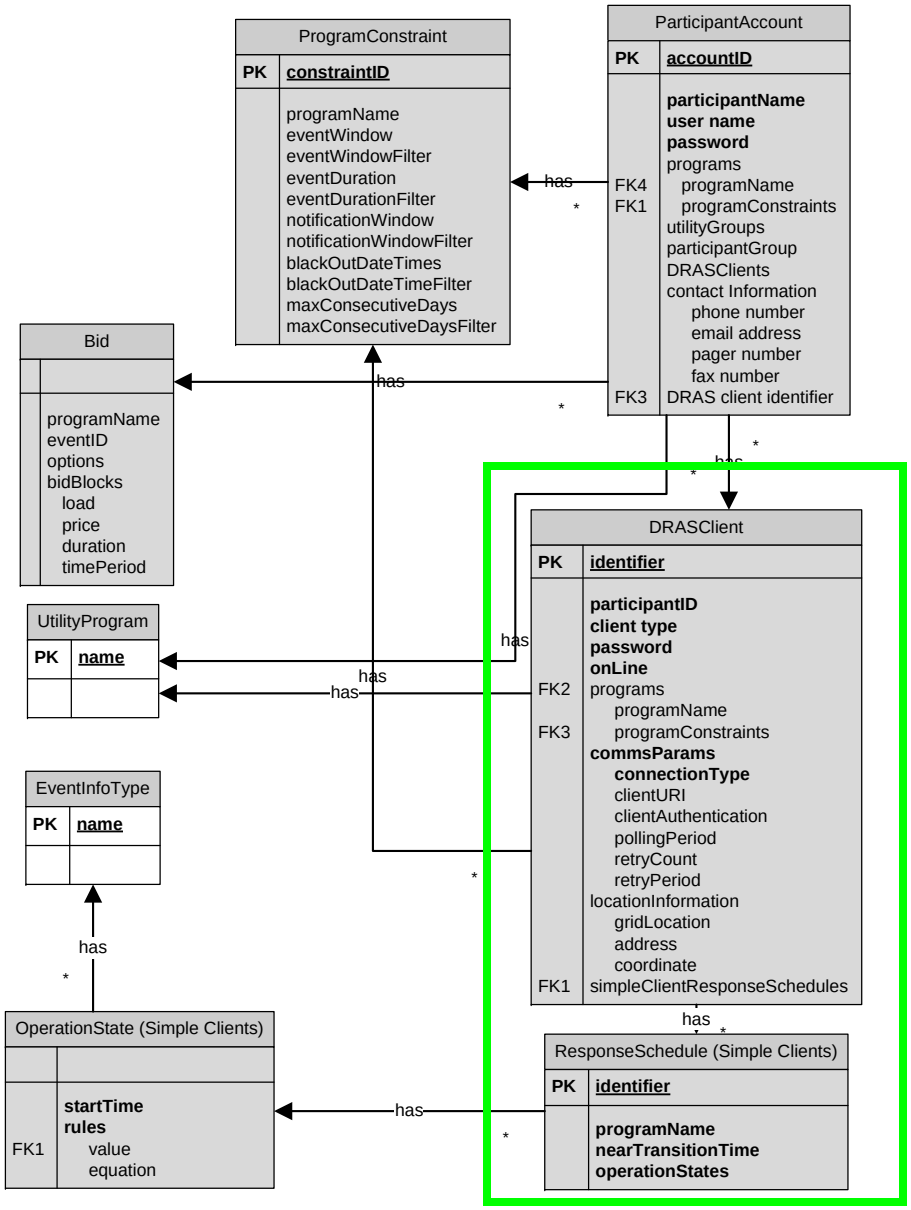




- Interaction between the DRAS and the DRAS Client involves:
  - ***EventState*** (DRAS Client ID, Program, Transaction ID, Simple & Smart DRAS Client data, Custom data, etc.)
  - ***EventStateConfirmation***



DRAS Client Interface



Participant Configuration Entities

- A ***ParticipantAccount*** may define ***DRASClient*** entities
- ***DRASClient*** entity represents DRAS Client information:
  - **Simple** or **Smart** type.
  - **Participating** Programs
  - DR ***ProgramConstraints***
  - **Communication** parameters.
  - **Location** information
  - [***ResponseSchedule*** entities]
- ***OperationStates*** define *ResponseSchedule* entity:
  - Ordered list of ***OperationStateSpec***

Methods & Their Functions

DRAS Client Interface

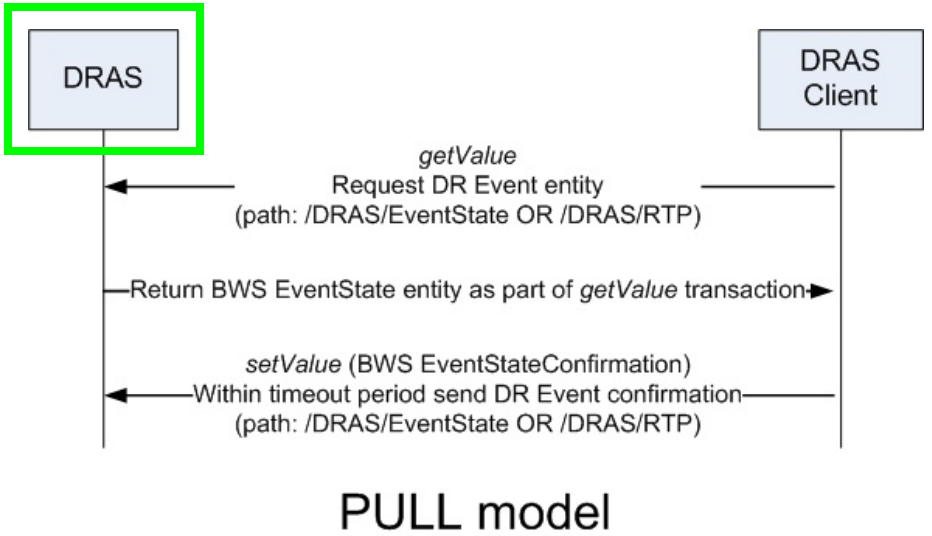
- Send DR Event Information (PUSH) to DRAS Client
  - *EventState* entity
- Get DR Event Information (PULL for DRAS Client)
  - *EventState* entity
- Send Event State Confirmation to DRAS
  - *EventStateConfirmation* entity

OpenADR & Control Protocols

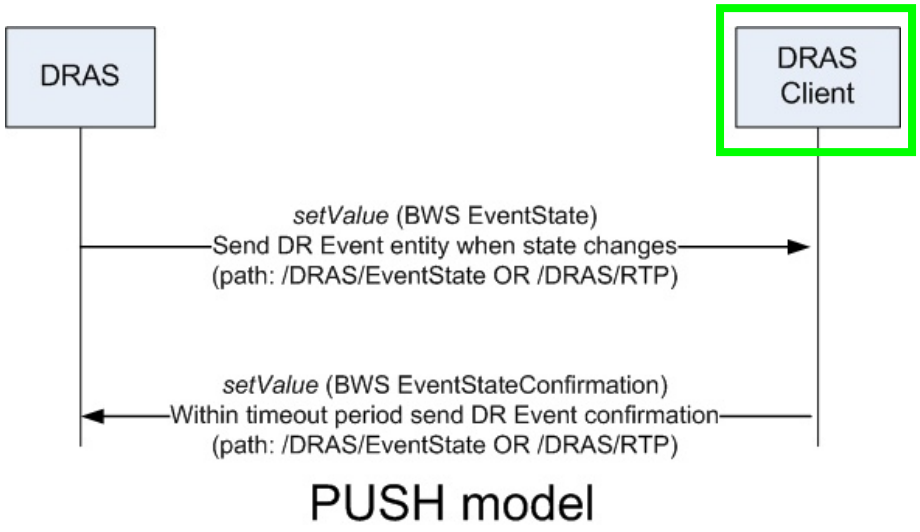
BACnet Web Services

DRAS-BACnet Server

- **Protocols Agnostic:** BACnet Web Services (BWS) specification: **ANSI/ ASHRAE 135-2004 Addendum C***
- BACnet – **UIWG** & **XMLWG**
- Control System Modeling Language:
  - Schema ***Definition*** and ***Instance*** XML
  - *EventState* Schema – R/W *String[]*
- BWS Services:
  - ***getValue*(R), *setValue*, *getDefaultLocale* (R), and *getSupportedLocals* (R).**
- **Node Tree-based** data model:
  - Tree Root: **/DRAS**
  - Tree Node: **/DRAS/EventState OR /DRAS/RTP**



PULL model



PUSH model

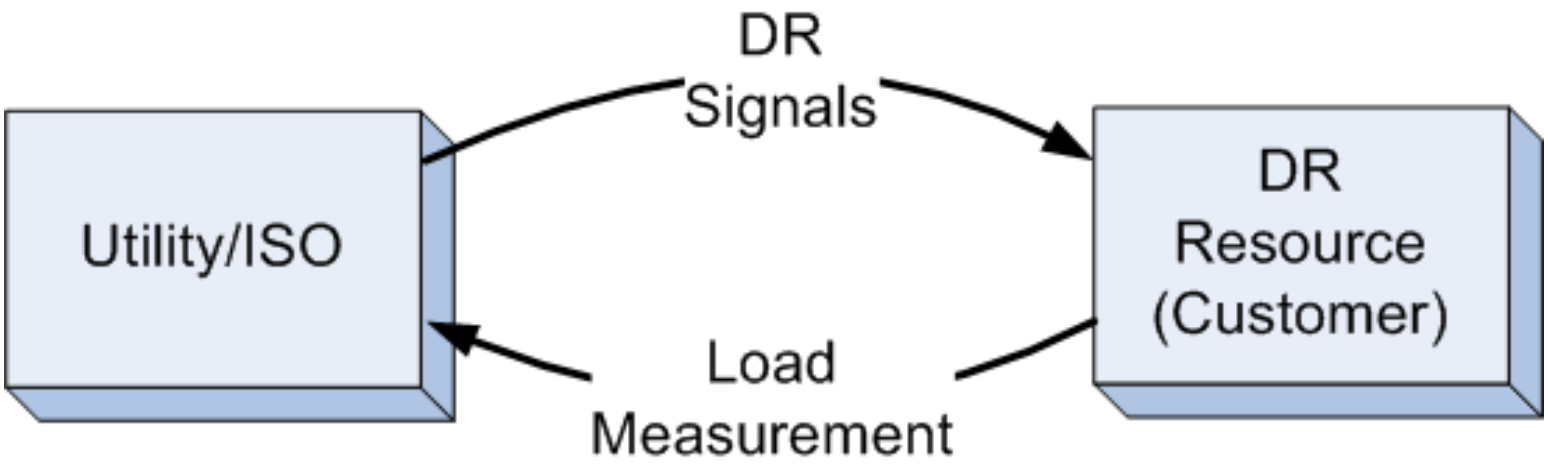
*** http://www.bacnet.org/Addenda/**

END OF SESSION 1

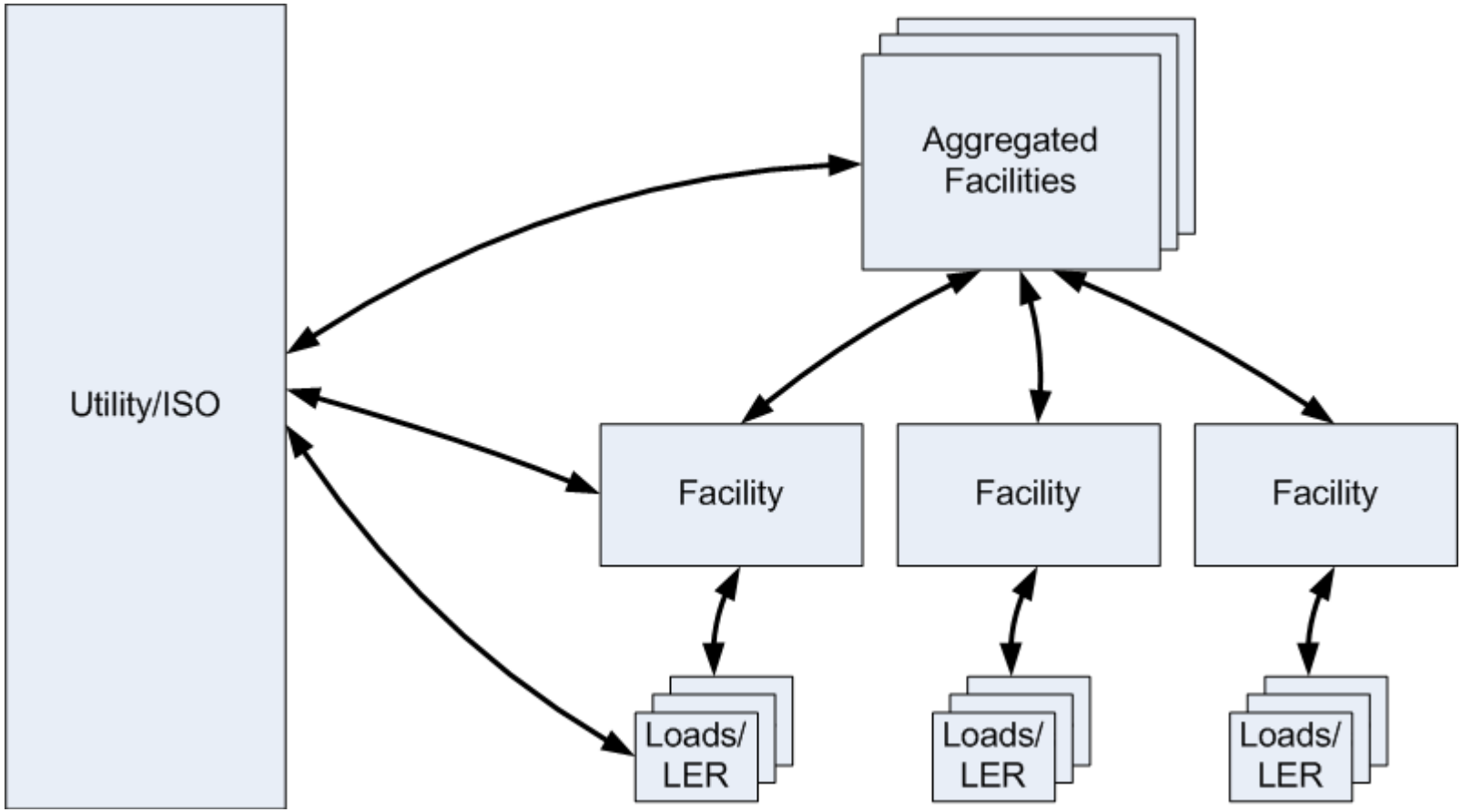
Wedesday, May 5, 10:30-12:00

- Confirm details of scope for SRS
- Discussion of general inter-domain interaction models to be used in SRS
  - Which entities?
  - Type of interactions that are in scope of SRS?

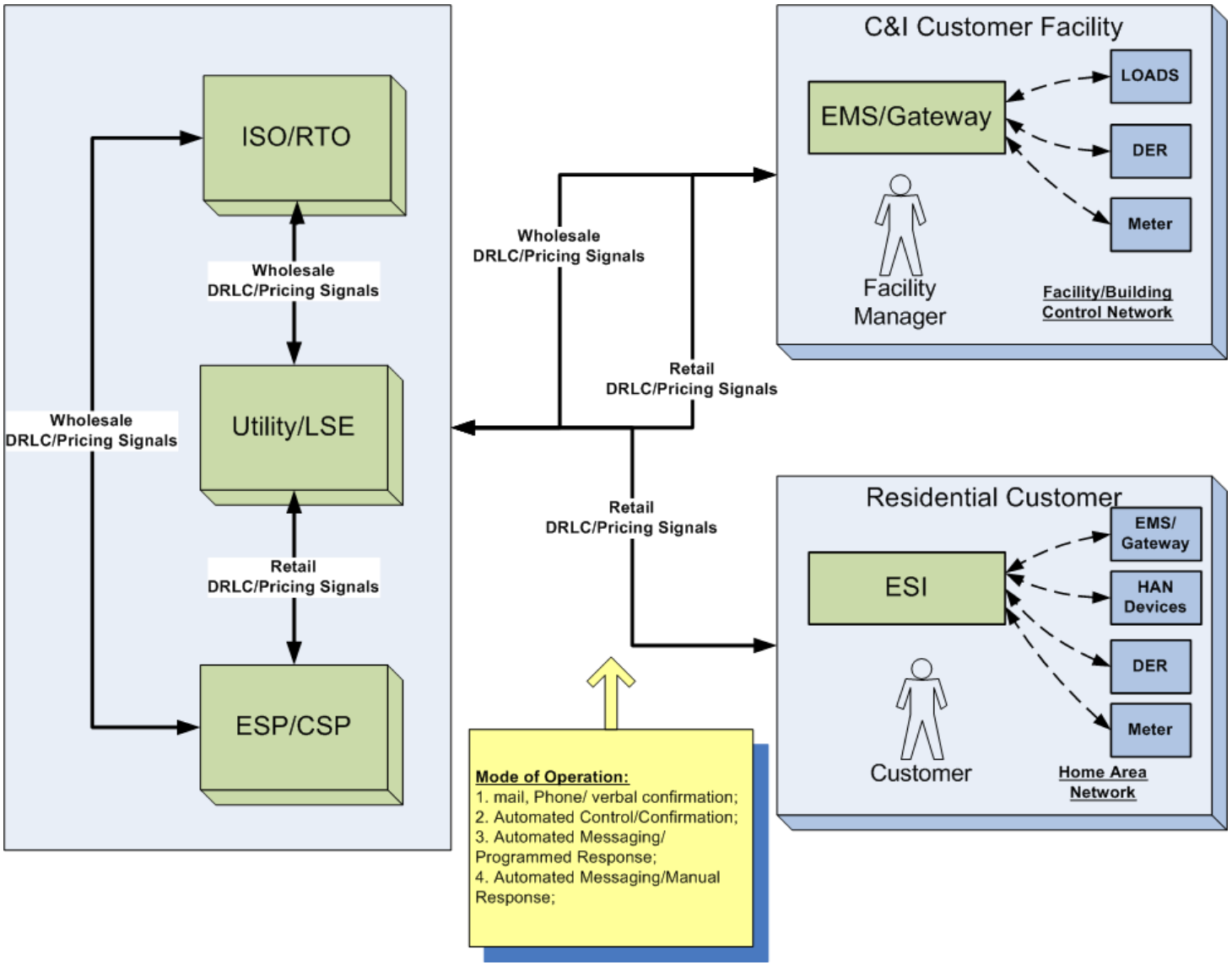
Simple Interaction Model

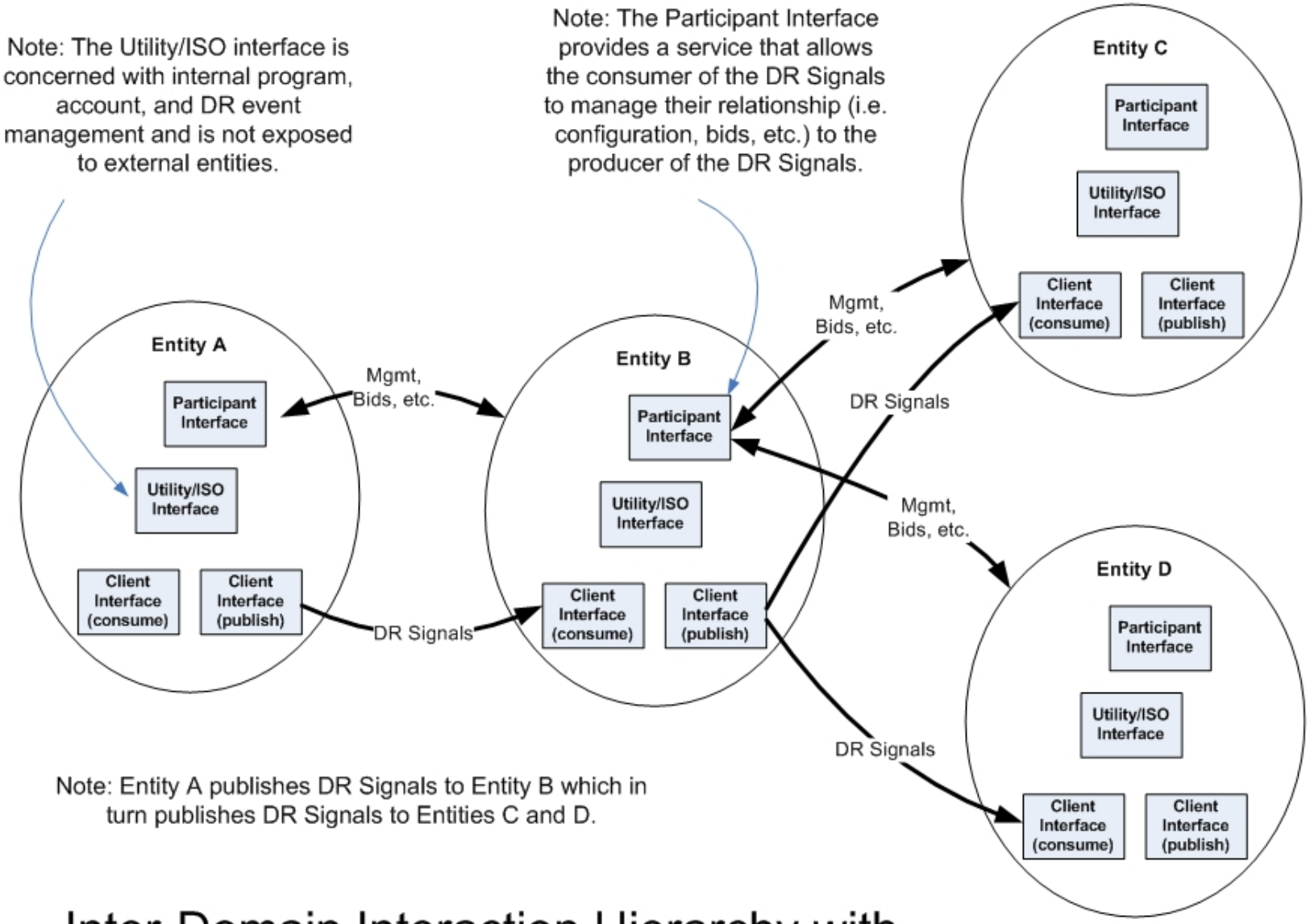


DR Resource Hierarchy



Interaction Diagram



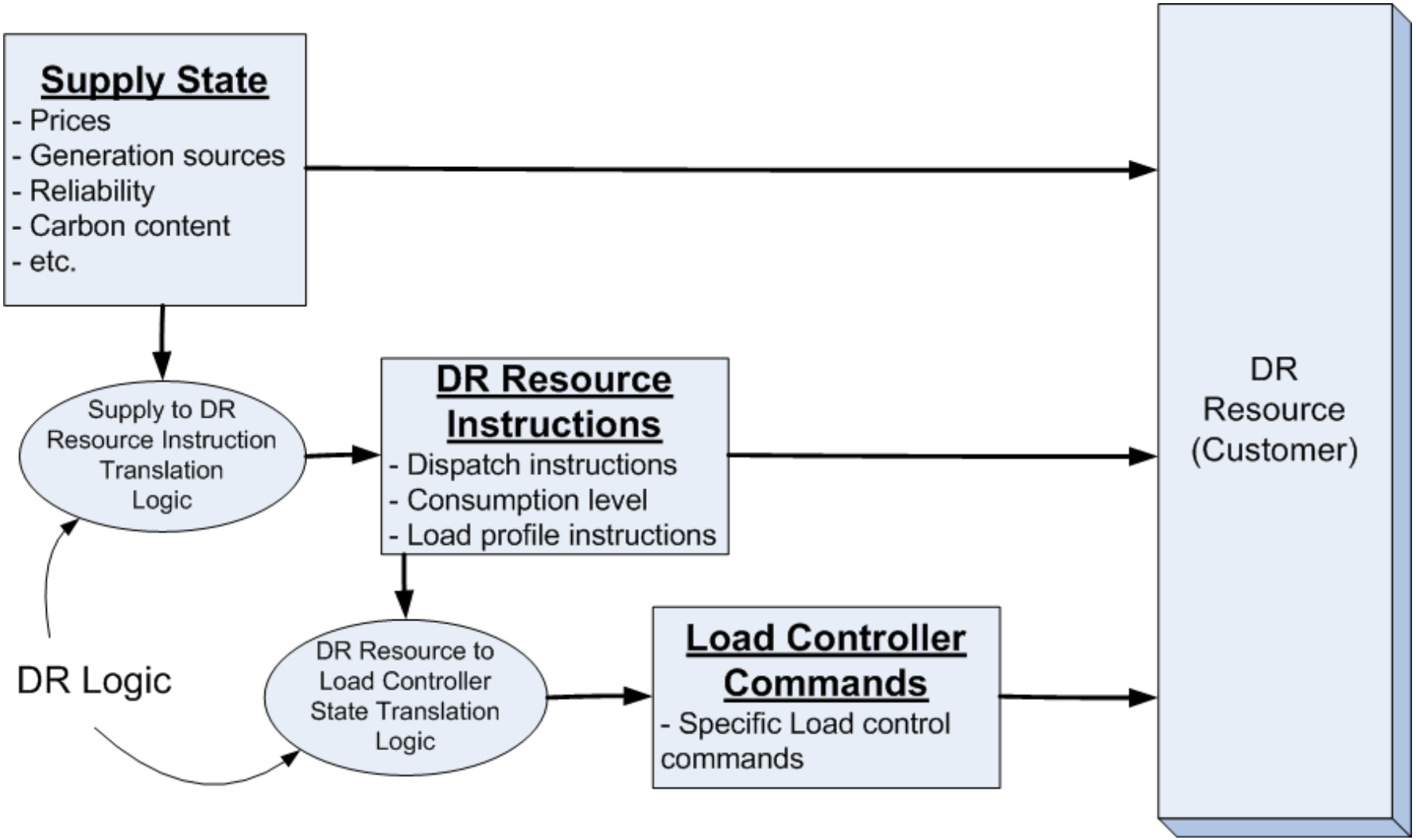


Inter-Domain Interaction Hierarchy with Respect to Current OpenADR Interfaces

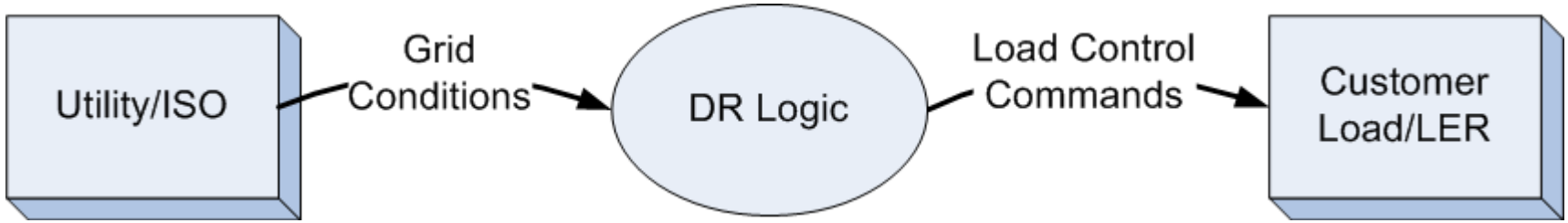
DR Signal Types

- **Supply State**
  - Prices
  - Generation sources
  - Reliability
  - Carbon content, etc.
- **DR Resource Instructions**
  - Dispatch instructions
  - Consumption level
  - Load profile instructions
- **Load Controller Commands**
  - Specific Load control commands

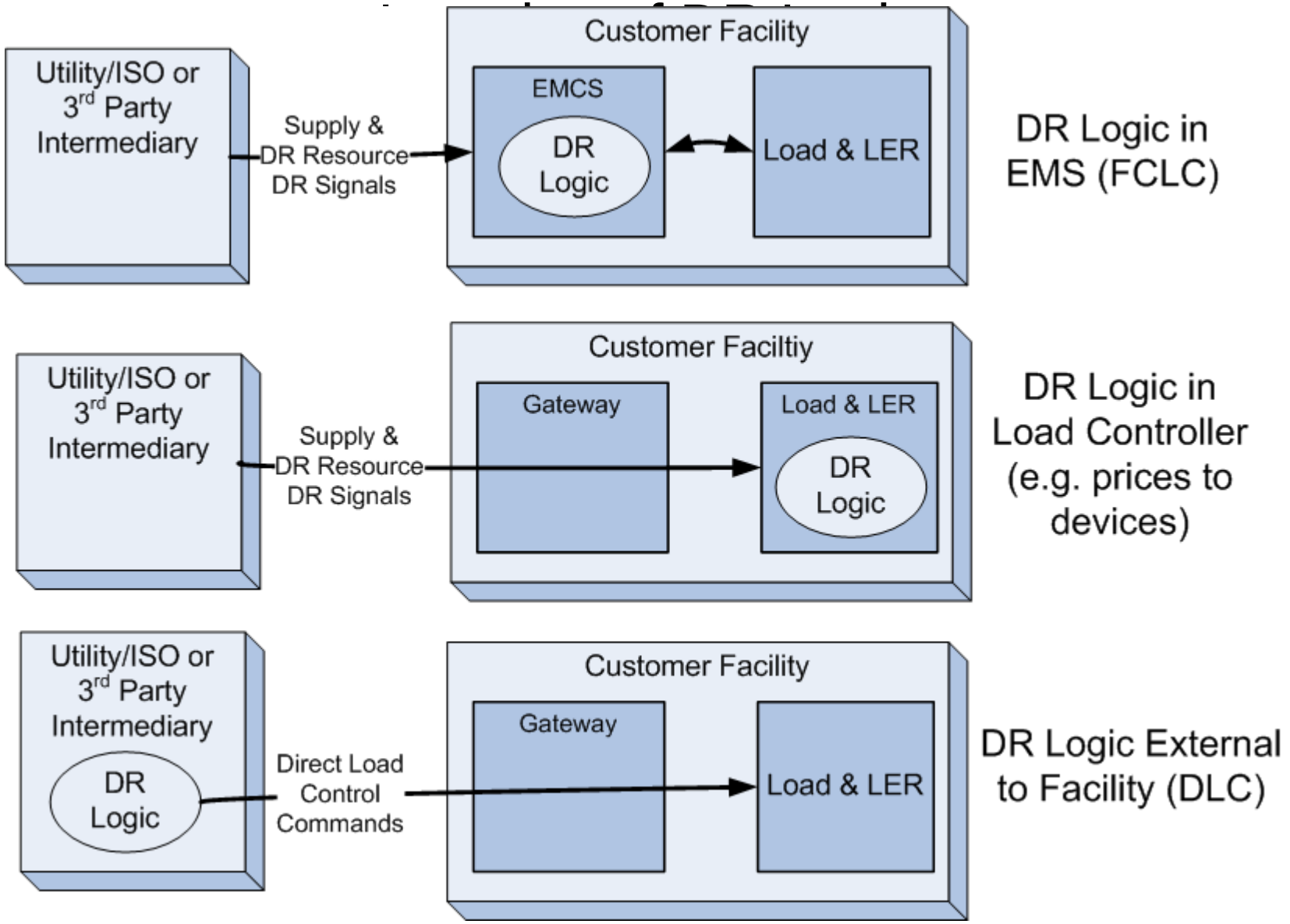
Interaction Mode Hierarchy



DR Logic Concept



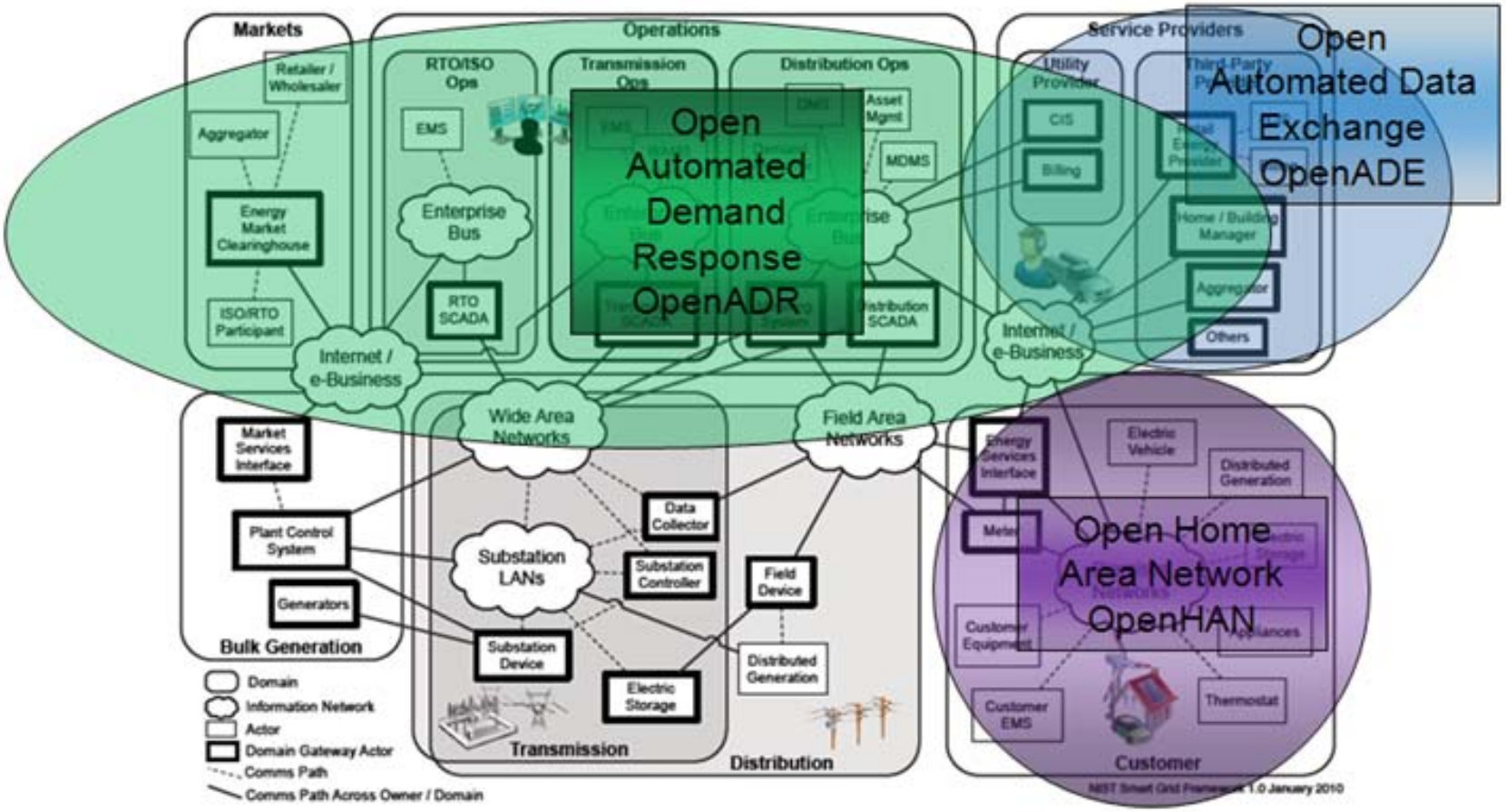
Definition of FCLC and DLC with Respect to



Wednesday, May 5, 3:30-5:30

- Utility enterprise integration and harmonization issues to be addressed in SRS
  - CIM, AMI-ENT, OpenADE, SEP

Consolidated Use Cases



OpenADE Use Cases

Thursday, May 6, 8:00-10:00

- Work on SRS document itself.
  - Review current draft, identify areas of focus