Appendix A

A. IMPLEMENTED INTERFACES IN IDL

In this appendix we provide all interfaces defined in IDL during the implementation of the proof-of-concepts. In some cases the files include interface methods that have not been implemented, these methods are defined without input or output parameters.

We first include the interfaces for the MANBoP components and the definition of the exceptions used. Then, we provide the CIA interfaces. In third place we provide the interfaces in IDL that define those Information Model Objects used in the proof-of-concepts. Finally, we have also included interfaces implemented to ease the use of the system. These tools are described in more detail in appendix D.

MANBoP component	*Interface in IDL*

Authorisation Check Component	#include <PCM.idl>
	module es{
	  module upc{
	    module nmg{
	      module MANBoP{
	        module AuthorisationCheckComponent{
	          interface ACnt{
	          boolean authorise(
	                            in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId,
	                            in es::upc::nmg::MANBoP::PolicyConsumerManager::credential User,
	                            in string domainId);
	          };
	        };
	      };
	    };
	  };
	};

	#include <PCM.idl>
	module es{
	  module upc{
	    module nmg{
	      module MANBoP{
	        module Database{
	          interface DBCore{
	          };
	          interface Policy{
	          boolean setPolicy(
	                            in es::upc::nmg::MANBoP::IM::User::Policy::Policy jpolicy,
	                            in string XPolicy,
	                            in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId,
	                            in es::upc::nmg::MANBoP::PolicyConsumerManager::credential User);
	          es::upc::nmg::MANBoP::IM::User::Policy::Policy getPolicy(
	                            in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId)
	                            raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
	          es::upc::nmg::MANBoP::PolicyConsumerManager::credential getPolicyUserCred(
	                            in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId)
	                            raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
	          string getXPolicy(
	                            in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId)
	                            raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
	          void getPolicies();
	          boolean modPStatus(
	                            in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId,
	                            in long value);
	          boolean removeP(
	                            in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId);
	          long getPSts(
	                            in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId)
	                            raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
	          boolean setPRI(
	                            in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId,
	                            in es::upc::nmg::MANBoP::IM::User::Policy::PRI pri);
	           es::upc::nmg::MANBoP::IM::User::Policy::PRI getPRI(
	                            in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId)
	                            raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
	           void rmPRI();
	          };
	
	          interface Schema{

```
boolean setSchema(
                in es::upc::nmg::MANBoP::IM::User::Policy::Schema sch,
                in string username,
                in string domainId);
es::upc::nmg::MANBoP::IM::User::Policy::Schema getSchema(
                in string username,
                in string domainId)
                raises (es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
void remSchema();
};
interface PGroup{
boolean setGroup(
                in es::upc::nmg::MANBoP::IM::User::Policy::Group group,
                in string username);
boolean setGroupP(
                in es::upc::nmg::MANBoP::IM::User::Policy::Policy jpolicy,
                in string XPolicy,
                in es::upc::nmg::MANBoP::PolicyConsumerManager::credential cred,
                in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId pid);
es::upc::nmg::MANBoP::IM::User::Policy::Group getGroup(
                in long groupnumber,
                in string username)
                raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
string getGroupXP(
                in long groupnumber,
                in string posId,
                in string username)
                raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
void getGroupSt();
es::upc::nmg::MANBoP::IM::User::Policy::Policy getGroupP(
                in long groupnumber,
                in string posId,
                in string username)
                raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
void modGroupSt();
boolean rmGroupP(
                in long groupnumber,
                in string posId,
                in string username);
boolean rmGroup(
                in long groupnumber,
                in string username);
es::upc::nmg::MANBoP::PolicyConsumerManager::credential getGroupPUserCred(
                in long groupnumber,
                in string posId,
                in string username)
                raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
es::upc::nmg::MANBoP::IM::User::Policy::t_policyId getGroupPId(
                in long groupnumber,
                in string posId,
                in string username)
                raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
};
typedef sequence<string> t_EEIdList;
typedef sequence<string> t_stringList;
interface Topology{
void getPath();
void createPath();
void modPath();
void rmPath();
boolean createGblTop(
                        in t_stringList nodes,
                        in t_stringList aps,
                        in t_stringList links);
IM::Topological::MgdTop::GblTop getGblTop()
                        raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
void modGblTop();
boolean createTopObj(
                        in string nodeId,
                        in long type,
                        in boolean edge,
                        in t_stringList outL,
                        in t_stringList inL,
                        in string nResoId,
                        in string nUResoId);
IM::Topological::MgdTop::Node getTopObj(
                        in string nodeId)
                        raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
void modTopObj();
void rmTopObj();
boolean createLinkObj(
                        in string linkId,
                        in string sourceNode,
                        in string sinkNode,
                        in long hops,
                        in long capacity,
                        in long uCapacity);
IM::Topological::MgdTop::Link getLinkObj(
                        in string nodeId,
                        in string linkId)
                        raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
boolean rmLinkObj();
boolean createNResoObj(
                        in string nResoId,
                        in long cpu,
                        in long disk,
                        in long memory,
                        in long numberOfEEs,
                        in t_EEIdList EEIds );
boolean getNResoObj()
                        raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
boolean rmNResoObj();
};

interface Manager{
IM::ManagerInstance::ManagerInstance getMI()
                        raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
boolean setMI(
                        in IM::ManagerInstance::ManagerInstance mi);
IM::ManagerInstance::Components::PCC getPCC()
                        raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
boolean setPCC(
                        in IM::ManagerInstance::Components::PCC pcc);
boolean createUndIntObj(
                        in string id,
                        in string iface,
                        in string nodeSetId,
                        in string nodeSetLoc,
                        in string addr,
                        in t_stringList info);
IM::ManagerInstance::UndInt::Device getUndIntObj(
                        in string nodeid)
                        raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
          boolean rmUndIntObj();
          IM::ManagerInstance::Components::PC getPC(
                        in string nodeSetid,
                        in string pcid)
                        raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
          boolean setPC(
                        in IM::ManagerInstance::Components::PC pc,
                        in string nodeSetId);
          boolean rmPC(
                        in string nodeSetid,
                        in string pcid);
          IM::ManagerInstance::Components::MM getMM(
                        in string nodeSetid,
                        in string mmid)
                        raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
          boolean setMM(
                        in IM::ManagerInstance::Components::MM mm,
                        in string nodeSetId);
          boolean rmMM(
                        in string nodeSetId,
                        in string mmid);
          };
          interface Resource{
          void createRI();
          void getRI();
          void modRI();
          void getUsedRI();
          void modUsedRI();
          void getRouteI();
          void modRouteI();
          };
          interface User{
          IM::User::User getUser(
                        in string username)
                        raises(es::upc::nmg::MANBoP::Exceptions::DBObjectNotFound);
          boolean setUser(
                        in IM::User::User user);
          };
        };
      };
    };
  };
};
```

Decision-making Monitoring system

```
#include <PCM.idl>
module es{
  module upc{
    module nmg{
      module MANBoP{
        module DmMs{
          interface DLgc{
          typedef sequence<string> t_nodeIds;
          boolean regCond(
                        in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId,
                        in es::upc::nmg::MANBoP::IM::User::Policy::Policy jpolicy,
                        in es::upc::nmg::MANBoP::PolicyConsumerManager::credential user);
          boolean unregCond(
                        in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId);
          boolean upUnI(
                        in boolean type,
                        in t_nodeIds undnodes);
        oneway void ISValue(
                           in string isid,
                           in boolean value);
          };
        };
      };
    };
  };
};
```

Monitoring Meter

```
#include <PCM.idl>
module es{
  module upc{
    module nmg{
      module MANBoP{
        module MonitoringMeter{
          typedef sequence<boolean> t_booleanList;
          interface MFact{
          t_booleanList monIS(
                           in string isid, in es::upc::nmg::MANBoP::IM::User::Policy::t_simpleCond sc,
                           in es::upc::nmg::MANBoP::PolicyConsumerManager::credential user)
                           raises (es::upc::nmg::MANBoP::Exceptions::MonitoringError);
          boolean sMonIS(
                           in string isid);
          oneway void uninstallYourself();
          void recvMonResult(
                           in string errorMessage,
                           in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId,
                           in boolean newvalue);
          };
        };
      };
    };
  };
};
```

Policy Consumer

```
#include <PCM.idl>
module es{
  module upc{
    module nmg{
      module MANBoP{
        module PolicyConsumer{
          typedef sequence<string> stringList;
          interface Mapper{
          es::upc::nmg::MANBoP::PolicyConsumerManager::Result enforceP(
                           in es::upc::nmg::MANBoP::IM::User::Policy::Policy jpolicy,
                           in long seqNumber,
                           in es::upc::nmg::MANBoP::PolicyConsumerManager::credential user,
                           in stringList nodes,
                           in long actionOrder)
                           raises (es::upc::nmg::MANBoP::Exceptions::UnableToEnforceP);
          void recvEnfResult(
                           in string errorMessage,
                           in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId,
                           in es::upc::nmg::MANBoP::PolicyConsumerManager::Result result);
          oneway void uninstallYourself();
          };
        };
      };
    };
  };
};
```

Policy Conflict Check

```
#include <PCM.idl>
module es{
   module upc{
     module nmg{
       module MANBoP{
         module PolicyConflictCheck{
           typedef sequence<string> t_stringList;
           interface PAn{
           boolean checkConfl(
                                   in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId);
           t_stringList checkDyn(
                                   in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId)
                                   raises (es::upc::nmg::MANBoP::Exceptions::UnProcessablePolicy);
           oneway void uninstPR(
                                   in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId,
                                   in boolean cause);
           };
         };
       };
     };
   };
};
```

Policy Consumer Manager

```
#include <IM.idl>
#include <Exceptions.idl>
module es{
   module upc{
     module nmg{
       module MANBoP{
         module PolicyConsumerManager{
           struct credential{
           string login;
           string passwd;
           };
           typedef sequence<string> stringList;
           struct Result{
           long value;
           stringList extraInfo;
           };
           interface PFwCnt{
           oneway void dispatch(
                                   in credential User,
                                   in string xpolicy)
                                   raises(es::upc::nmg::MANBoP::Exceptions::UnProcessablePolicy);
           void pProcSt(
                                   in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId,
                                   in Result result,
                                   in string error);
           };
           interface PCMCore{
           struct decision{
           boolean is_accepted;
           string reason;
           };
           void main(
                                   in long mgmtTopId,
                                   in string MgdTop_file,
                                   in string UndInt_file);
           void addN(
                                   in string MgdTop_file,
                                   in string UndInt_file);
```

```
          void removeN(
                      in string MgdTop_file);
          void procP(
                      in credential User,
                      in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId)
                      raises(es::upc::nmg::MANBoP::Exceptions::UnProcessablePolicy);
          void procS();
          oneway void triggerEnf(
                      in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId,
                      in boolean reason);
          boolean uninstallP(
                      in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId,
                      in long reason);
          };
        };
      };
    };
  };
};
```

Policy Editor

```
#include <PC.idl>
module es{
  module upc{
    module nmg{
      module MANBoP{
        module PolicyEditor{
          interface PECore{
          typedef sequence<string> t_pServicesList;
          typedef sequence<string> t_oidList;
          t_pServicesList registerUser(
                      in es::upc::nmg::MANBoP::PolicyConsumerManager::credential user);
          t_oidList rqPServices(
                      in t_pServicesList pServices);
          void policyInfo(
                      in string info,
                      in es::upc::nmg::MANBoP::PolicyConsumerManager::credential user);
          void policyInfo_(
                      in string rqId,
                      in string info,
                      in es::upc::nmg::MANBoP::PolicyConsumerManager::credential user);
          oneway void recvXPolicy(
                      in es::upc::nmg::MANBoP::PolicyConsumerManager::credential user,
                      in string XPolicy)
                      raises(es::upc::nmg::MANBoP::Exceptions::UnknownUser);
          oneway void recvEnfResult(
                      in string errorMessage,
                      in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId,
                      in es::upc::nmg::MANBoP::PolicyConsumerManager::Result result);
          };
        };
      };
    };
  };
};
```

SigDemux

```
module es{
  module upc{
    module nmg{
      module MANBoP{
        module SigDemux{
          interface PCDmx{
          struct credential{
          string login;
```

	string passwd;
	};
	typedef sequence<string> t_filterIds;
	typedef sequence<string> t_filters;
	boolean regPC(
	in string pcid,
	in string pcint,
	in t_filterIds fids,
	in t_filters fs);
	void recvRq(
	in string filterId,
	in string request,
	in string nodeAddr);
	};
	interface Listener{
	boolean setFilter(
	in string filterId,
	in string filter);
	};
	};
	};
	};
	};
	};

Traffic Engineering Manager	module es{
	module upc{
	module nmg{
	module MANBoP{
	module TEManager{
	interface TECore{
	struct flow{
	string srcIP;
	string destIP;
	string srcPort;
	string destPort;
	string Prot;
	};
	typedef sequence<string> t_tdayList;
	struct interval{
	long start_time;
	short moymask;
	long dommask;
	octet dowmask;
	t_tdayList tday;
	long end_time;
	};
	struct resource{
	long resourceType;
	string resourceId;
	interval enforcementInt;
	long reqValue;
	};
	typedef sequence<string> t_nodeIds;
	typedef sequence<resource> t_resources;
	boolean findRoute(
	in flow assFlow,
	in t_resources res,
	in string endPointA,
	in string endPointZ,
	in boolean directionality,
	in boolean type,

```
                                        out t_nodeIds nodeids,
                                        out string pathId,
                                        out t_resources cres);
                    boolean estimateCosts();
                    boolean updateTop(
                                        in boolean type);
                };
            };
        };
      };
    };
};
```

Table A - 1. MANBoP interfaces in IDL

The following table shows the definition in IDL of the exceptions used in MANBoP.

IDL definition of MANBoP Exceptions

```
module es{
    module upc{
        module nmg{
            module MANBoP{
                module Exceptions{
                    typedef sequence<string> t_stringList;
                    exception DBObjectNotFound {
                        string detail;
                    };
                    exception UnknownUser{
                        string detail;
                    };
                    exception UnProcessablePolicy{
                        string detail;
                    };
                    exception UnableToEnforceP{
                        string detail;
                        t_stringList tn_OK;
                        t_stringList tn_KO;
                    };
                    exception MonitoringError{
                        string detail;
                    };
                };
            };
        };
    };
};
```

Table A - 2. MANBoP exceptions

The next table includes the definition in IDL of the CIA component interfaces; both for the server and client side.

CIA component in IDL

```
module org{
  module corba{
      module utils{
          module CIA{
              typedef sequence<string> t_stringList;
              exception CodeNotFound{
                     string detail;
              };
              interface CIA{
                     void dwCode(
                                     in string codeId,
                                     in string version,
                                     in t_stringList args)
                                     raises (CodeNotFound);
                     void dwSchema(
                                     in string schemaId,
                                     in string dest)
                                     raises (CodeNotFound);
              };
              interface CodeServer{
                     string obtainLocation(
                                     in string codeId,
                                     in string version,
                                     in long type)
                                     raises (CodeNotFound);
                     void obtainCode(
                                     in string location);
              };
          };
      };
  };
};
```

Table A - 3. CIA component interfaces

Those IMOs implemented for the proof-of-concepts are included in the table below.

IMOs in IDL

```
module es{
  module upc{
    module nmg{
      module MANBoP{
        module IM{
          module Topological{
            module MgdTop{
              typedef sequence<string> t_stringList;
              struct GblTop{
                        t_stringList nodes;
                        t_stringList aps;
                        t_stringList links;
              };
              struct Node{
                        string nodeId;
                        long type;
                        boolean edge;
                        t_stringList outL;
                        t_stringList inL;
                        string nResoId;
                        string nUResoId;
              };
              struct Link{
                        string linkId;
                        string sourceNode;
                        string sinkNode;
                        long hops;
                        long capacity;
                        long uCapacity;
              };
              struct NResources{
                        string nResoId;
                        long cpu;
                        long disk;
                        long memory;
                        long numberOfEEs;
                        t_stringList EEIds;
              };
              struct UNResources{
                        string uNResoId;
                        long cpu;
                        long disk;
                        long memory;
                        long numberOfEEs;
                        t_stringList EEIds;
              };
            };
          };
          module ManagerInstance{
            typedef sequence<string> t_stringList;
            struct ManagerInstance{
              string managerId;
              t_stringList pcs;
              t_stringList mms;
              string pcc;
              t_stringList devices;
            };
            module Components{
              struct Component{
                        string componentId;
```

```
                string version;
            };

            struct PCC{
                string componentId;
                string version;
                t_stringList suppDomains;
            };

            struct PC{
                string pcId;
                string version;
                long mgmtTopId;
                string iface;
            };

            struct MM{
                string mmId;
                string version;
                long mgmtTopId;
                string iface;
            };
        };
        module UndInt{
            struct Device{
                string id;
                string iface;
                string nodeSetId;
                string nodeSetLoc;
                string addr;
                t_stringList info;
            };
        };
    };
    module User{
        struct User{
            string username;
            string password;
        };
        module Policy{
            typedef sequence<string> t_stringList;
            struct t_policyruleId{
                string prname;
                long sequenceNumber;
                string hlpcid;
            };
            struct t_policyId{
                t_policyruleId ruleId;
                string username;
                string domainId;
                string condNames;
            };
            struct t_userInfo{
                string username;
                string pass;
            };
            struct t_period{
                string startD;
                string startT;
                string endD;
                string endT;
```

```
};
struct t_prvp{
        t_period period;
        short moymask;
        long long dommask;
        octet dowmaks;
        string timeofday;
        boolean localOrUTCtime;

};
struct t_split{
        long subpos;
        long subnumber;
};
typedef sequence<t_split> t_splitList;
struct t_order{
        string posId;
        long position;
        long numSplits;
        t_splitList pos;
};
struct t_pgroup{
        long pgnum;
        long nofp;
        long execst;
        long where;
        t_order pos;
};
struct t_eval{
        boolean cl;
        long where;
};
struct t_simpleCond{
        string pcn;
        long groupnumber;
        boolean condnegated;
        string MMId;
        t_stringList tn;
        boolean all;
        string dataName;
        string dataType;
        string evalmethod;
        string value;
};
typedef sequence<t_simpleCond> t_scList;
struct t_comCond;
typedef sequence<t_comCond> t_ccList;
struct t_comCond{
        string pcn;
        long groupnumber;
        boolean condnegated;
        t_stringList mmids;
        boolean ismirrored;
        t_scList scondL;
        t_ccList ccondL;
};
struct t_condRef{
        t_scList scondL;
        t_ccList ccondL;

};
```

```
                                    struct t_actRef{
                                          string actType;
                                          string name;
                                          string pcId;
                                          t_stringList attrnames;
                                          t_stringList attrvalues;
                                    };
                                    typedef sequence<t_condRef> t_crefList;
                                    typedef sequence<t_actRef> t_arefList;
                                    struct actEnf{
                                          long actMode;
                                          long enfType;
                                          boolean ms;
                                          t_stringList tn;
                                    };
                                    struct Policy{
                                          string schemaId;
                                          t_policyruleId ruleId;
                                          long status;
                                          t_stringList roles;
                                          t_userInfo user;
                                          t_prvp validity;
                                          t_pgroup group;
                                          t_eval evaluation;
                                          actEnf act;
                                          t_crefList conds;
                                          t_arefList actions;
                                    };
                                    typedef sequence<t_order> t_orderList;
                                    struct Group{
                                          long pgnum;
                                          long nofp;
                                          long execst;
                                          t_orderList recvPs;
                                          t_orderList sendPs;
                                          t_orderList enfPs;
                                          t_orderList remPs;
                                          long status;
                                    };
                                    struct Schema{
                                          string schemaId;
                                    };
                                    struct PRI{
                                          t_stringList nodes;
                                    };
                              };
                        };
                  };
            };
      };
   };
};
```

Table A - 4. MANBoP IMOs

The last table included in this appendix shows the definition in IDL of generic interface that must be implemented to receive enforcement results from the MANBoP framework.

Generic interface in IDL

```
#include <PC.idl>
module org{
  module corba{
     module utils{
        interface GenRecv{
           void recvEnfResult(
                       in string errorMessage,
                       in es::upc::nmg::MANBoP::IM::User::Policy::t_policyId policyId,
                       in es::upc::nmg::MANBoP::PolicyConsumerManager::Result result);
        };
     };
  };
};
```

Table A - 5. Generic interface