NORME INTERNATIONALE

INTERNATIONAL STANDARD

CEI
IEC
61970-1

Première édition
First edition
2005-12

Interface de programmation d'application pour système de gestion d'énergie (EMS-API) –

Partie 1: Lignes directrices et exigences générales

Energy management system application program interface (EMS-API) –

Part 1: Guidelines and general requirements

Numéro de référence
Reference number
CEI/IEC 61970-1:2005

Numérotation des publications

Depuis le 1er janvier 1997, les publications de la CEI sont numérotées à partir de 60000. Ainsi, la CEI 34-1 devient la CEI 60034-1.

Editions consolidées

Les versions consolidées de certaines publications de la CEI incorporant les amendements sont disponibles. Par exemple, les numéros d'édition 1.0, 1.1 et 1.2 indiquent respectivement la publication de base, la publication de base incorporant l'amendement 1, et la publication de base incorporant les amendements 1 et 2.

Informations supplémentaires sur les publications de la CEI

Le contenu technique des publications de la CEI est constamment revu par la CEI afin qu'il reflète l'état actuel de la technique. Des renseignements relatifs à cette publication, y compris sa validité, sont disponibles dans le Catalogue des publications de la CEI (voir ci-dessous) en plus des nouvelles éditions, amendements et corrigenda. Des informations sur les sujets à l'étude et l'avancement des travaux entrepris par le comité d'études qui a élaboré cette publication, ainsi que la liste des publications parues, sont également disponibles par l'intermédiaire de:

- **Site web de la CEI (www.iec.ch)**
- **Catalogue des publications de la CEI**

  Le catalogue en ligne sur le site web de la CEI (www.iec.ch/searchpub) vous permet de faire des recherches en utilisant de nombreux critères, comprenant des recherches textuelles, par comité d'études ou date de publication. Des informations en ligne sont également disponibles sur les nouvelles publications, les publications remplacées ou retirées, ainsi que sur les corrigenda.
- **IEC Just Published**

  Ce résumé des dernières publications parues (www.iec.ch/online_news/justpub) est aussi disponible par courrier électronique. Veuillez prendre contact avec le Service client (voir ci-dessous) pour plus d'informations.
- **Service clients**

  Si vous avez des questions au sujet de cette publication ou avez besoin de renseignements supplémentaires, prenez contact avec le Service clients:

  Email: custserv@iec.ch
  Tél: +41 22 919 02 11
  Fax: +41 22 919 03 00

Publication numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series. For example, IEC 34-1 is now referred to as IEC 60034-1.

Consolidated editions

The IEC is now publishing consolidated versions of its publications. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Further information on IEC publications

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology. Information relating to this publication, including its validity, is available in the IEC Catalogue of publications (see below) in addition to new editions, amendments and corrigenda. Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is also available from the following:

- **IEC Web Site (www.iec.ch)**
- **Catalogue of IEC publications**

  The on-line catalogue on the IEC web site (www.iec.ch/searchpub) enables you to search by a variety of criteria including text searches, technical committees and date of publication. On-line information is also available on recently issued publications, withdrawn and replaced publications, as well as corrigenda.
- **IEC Just Published**

  This summary of recently issued publications (www.iec.ch/online_news/justpub) is also available by email. Please contact the Customer Service Centre (see below) for further information.
- **Customer Service Centre**

  If you have any questions regarding this publication or need further assistance, please contact the Customer Service Centre:

  Email: custserv@iec.ch
  Tel: +41 22 919 02 11
  Fax: +41 22 919 03 00

.

NORME INTERNATIONALE

INTERNATIONAL STANDARD

CEI
IEC
61970-1

Première édition
First edition
2005-12

Interface de programmation d'application pour système de gestion d'énergie (EMS-API) –

Partie 1: Lignes directrices et exigences générales

Energy management system application program interface (EMS-API) –

Part 1: Guidelines and general requirements

International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland
Telephone: +41 22 919 02 11 Telefax: +41 22 919 03 00 E-mail: inmail@iec.ch Web: www.iec.ch

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

CODE PRIX
PRICE CODE X

Pour prix, voir catalogue en vigueur
For price, see current catalogue

SOMMAIRE

CONTENTS

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

INTERFACE DE PROGRAMMATION D'APPLICATION POUR SYSTÈME DE GESTION D'ÉNERGIE (EMS-API) –

Partie 1: Lignes directrices et exigences générales

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (CEI) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, la CEI – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de la CEI"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec la CEI, participent également aux travaux. La CEI collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de la CEI concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de la CEI intéressés sont représentés dans chaque comité d'études.
3) Les Publications de la CEI se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de la CEI. Tous les efforts raisonnables sont entrepris afin que la CEI s'assure de l'exactitude du contenu technique de ses publications; la CEI ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de la CEI s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de la CEI dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de la CEI et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) La CEI n'a prévu aucune procédure de marquage valant indication d'approbation et n'engage pas sa responsabilité pour les équipements déclarés conformes à une de ses Publications.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à la CEI, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de la CEI, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de la CEI ou de toute autre Publication de la CEI, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de la CEI peuvent faire l'objet de droits de propriété intellectuelle ou de droits analogues. La CEI ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de propriété et de ne pas avoir signalé leur existence.

La Norme Internationale CEI 61970-1 a été préparée par le comité technique CEI 57: Gestion des systèmes de puissance et échanges d'informations associés.

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
57/777/FDIS	57/795/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

Cette publication a été rédigée selon les Directives ISO/CEI, Partie 2.

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ENERGY MANAGEMENT SYSTEM APPLICATION PROGRAM INTERFACE (EMS-API) –

Part 1: Guidelines and general requirements

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with an IEC Publication.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61970-1 has been prepared by IEC technical committee 57: Power systems management and associated information exchange.

The text of this standard is based on the following documents:

FDIS	Report on voting
57/777/FDIS	57/795/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

La CEI 61970 comprend les parties suivantes, sous le titre général *Interface de programmation d'application pour système de gestion d'énergie (EMS-API:*

Partie 1: Lignes directrices et exigences générales

Partie 2: Glossary

Partie 301: Base de modèle d'information commun (CIM)

Partie 302: Common information model (CIM) financial, energy scheduling and reservations [1]

Partie 401: Spécification d'interface des composants – Cadre général

Partie 402: Component interface specification (CIS) – Common services[1]

Partie 403: Component Interface Specification (CIS) – Generic data access[1]

Partie 404: Component Interface Specification (CIS) – High speed data access[1]

Partie 405: Component Interface Specification (CIS) – Generic eventing and subscription[1]

Partie 407: Component Interface Specification (CIS) – Time series data access[1]

Partie 453: Exchange of Graphics Schematics Definitions (Common Graphics Exchange)[1]

Partie 501: Common Information Model (CIM) XML Codification for Programmable Reference and Model Data Exchange

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de maintenance indiquée sur le site web de la CEI sous «http://webstore.iec.ch» dans les données relatives à la publication recherchée. A cette date, la publication sera

- reconduite;
- supprimée;
- remplacée par une édition révisée, ou
- amendée.

1 A l'étude.

IEC 61970 consists of the following parts, under the general title *Energy management system application program interface (EMS-API)*:

Part 1: Guidelines and general requirements

Part 2: Glossary

Part 301: Common Information Model (CIM) base

Part 302: Common information model (CIM) financial, energy scheduling and reservations[1]

Part 401: Component interface specification (CIS) framework

Part 402: Component interface specification (CIS) – Common services[1]

Part 403: Component Interface Specification (CIS) – Generic data access[1]

Part 404: Component Interface Specification (CIS) – High speed data access[1]

Part 405: Component Interface Specification (CIS) – Generic eventing and subscription[1]

Part 407: Component Interface Specification (CIS) – Time series data access[1]

Part 453: Exchange of Graphics Schematics Definitions (Common Graphics Exchange)[1]

Part 501: Common Information Model (CIM) XML Codification for Programmable Reference and Model Data Exchange

The committee has decided that the contents of this publication will remain unchanged until the maintenance result date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed;
- withdrawn;
- replaced by a revised edition, or
- amended.

[1] Under consideration.

INTRODUCTION

La présente Norme fait partie de la série CEI 61970 qui définit les interfaces de programmation d'application (API) pour un système de gestion d'énergie (EMS). La présente norme se fonde en grande partie sur les travaux réalisés dans le cadre du projet de recherche (RP-3654-1) sur les API de centres de commande (CCAPI) de l'EPRI. Le projet CCAPI de l'EPRI a principalement pour objet de:

– réduire les coûts et les délais nécessaires à l'ajout de nouvelles applications à un EMS ou à un autre système[2];

– protéger l'investissement réalisé dans des applications actuellement existantes et fonctionnant efficacement;

– améliorer la capacité d'échange d'informations entre des systèmes distincts situés à l'intérieur ou à l'extérieur de l'environnement du centre de conduite.

L'approche technique doit fournir un cadre général d'intégration pour l'interconnexion d'applications/systèmes existants qui soit:

– fondé sur une architecture et un modèle d'information communs;

– indépendant de la technologie sous-jacente.

La série de normes 61970 a pour principal objet d'élaborer un ensemble de lignes directrices et de normes visant à faciliter 1) l'intégration d'applications développées par différents fournisseurs dans l'environnement du centre de conduite [3] et 2) l'échange d'informations avec des systèmes externes à l'environnement du centre de conduite. Le domaine d'application de ces spécifications couvre d'autres systèmes de transmission ainsi que des systèmes de distribution et de production externes au centre de conduite et appelés à échanger en temps réel des données opérationnelles avec le centre de conduite. De ce fait, lesdites normes ont également pour objet de permettre l'intégration de systèmes hérités existants et de nouveaux systèmes construits en conformité avec les normes relatives à ces domaines d'application.

L'ensemble complet des documents normatifs est constitué des parties suivantes:

Partie 1: Lignes directrices et exigences générales

Partie 2: Glossaire

Partie 3xx: Modèle d'information commun (CIM)

Partie 4xx: Spécifications d'interface de composants (CIS)

Partie 5xx: Mise en correspondance des technologies CIS

La CEI 61970-1 fournit un ensemble de lignes directrices et de capacités d'infrastructure générales nécessaires à l'application des normes d'interface EMS-API. Le présent document décrit le modèle de référence fournissant le cadre général d'application des autres parties des normes EMS-API. Le modèle de référence est fondé sur une architecture à composants de sorte que les normes portent sur les interfaces des composants pour l'échange d'informations entre des applications dans un environnement de centre de conduite. Ce modèle peut également s'appliquer à des échanges d'informations similaires entre des applications et des systèmes externes à l'environnement de centre de conduite, tels que d'autres centres de commande, des exploitants de réseau autonomes (ERA), des organisations régionales de transport (ORT) et des systèmes de gestion de la distribution (SGD).

La CEI 61970-1 comporte également des capacités générales pour l'infrastructure d'intégration qui, bien qu'elle soit exclue de cette partie de la norme, est destinée à fournir certains services essentiels à l'appui des normes d'interface EMS-API.

---

2 Dans l'idéal, il convient que l'installation d'une application sur un système requière un effort minimal et aucune modification du code source, de manière similaire au mode d'installation de progiciels sur un ordinateur de bureau. Le projet EMSAPI vise tout du moins à s'approcher de cet idéal en réduisant les efforts souvent considérables actuellement requis pour installer des applications tierces dans un EMS.

3 L'environnement du centre de commande comprend une commande de transmission conventionnelle au sein d'une entreprise de service public ainsi que les plus récents exploitants de réseau autonomes (ERA) et les exploitants régionaux de transport qui ne sont affiliés à aucune entreprise de service public.

INTRODUCTION

This standard is part of the IEC 61970 series that defines application program interfaces (APIs) for an energy management system (EMS). This standard is based to a large extent upon the work of the EPRI Control Center API (CCAPI) research project (RP-3654-1). The principle objectives of the EPRI CCAPI project are to:

– reduce the cost and time needed to add new applications to an EMS or other system[2];

– protect the investment in existing applications that are working effectively;

– improve the capability to exchange information between disparate systems both within and external to the control center environment.

The technical approach is to provide an integration framework for interconnecting existing applications/systems that is

– based on a common architecture and information model;

– independent of the underlying technology.

The principal task of the IEC 61970 series of standards is to develop a set of guidelines and standards to facilitate 1) the integration of applications developed by different suppliers in the control center environment[3] and 2) the exchange of information to systems external to the control center environment. The scope of these specifications includes other transmission systems as well as distribution and generation systems external to the control center that need to exchange real-time operational data with the control center. Therefore, another related goal of these standards is to enable the integration of existing legacy systems as well as new systems built to conform to these standards in these application domains.

The complete set of standards includes the following parts:

Part 1: Guidelines and general requirements

Part 2: Glossary

Part 3XX: Common Information Model (CIM)

Part 4XX: Component Interface Specification (CIS)

Part 5XX: CIS Technology Mappings

IEC 61970-1 provides a set of guidelines and general infrastructure capabilities needed for the application of the EMS-API interface standards. It describes the reference model that provides the framework for the application of the other parts of the EMS-API standards. This reference model is based on a component architecture, which places the focus of the standards on component interfaces for information exchange between applications in a control center environment. The model is also applicable to similar information exchanges between control center applications and systems external to the control center environment, such as other control centers, Independent System Operators (ISOs), Regional Transmission Organizations (RTOs), and Distribution Management Systems (DMS).

IEC 61970-1 also includes general capabilities for the integration infrastructure, which while not part of this standard, is expected to provide certain essential services to support the EMS-API interface standards.

---

2 Ideally, an application should be installed on a system with minimal effort and no modification of source code; i.e., the way software packages are installed on a desktop computer. The EMS-API Project goal is to at least approach that ideal by reducing the often significant efforts currently required to install third-party applications in an EMS.

3 The control center environment includes traditional transmission control within a utility as well as the newer Independent System Operators (ISOs) and Regional Transmission Operators (RTOs), which are not affiliated with any one utility.

INTERFACE DE PROGRAMMATION D'APPLICATION POUR SYSTÈME DE GESTION D'ÉNERGIE (EMS-API) –

Partie 1: Lignes directrices et exigences générales

1 Domaine d'application

La présente partie de la série CEI 61970 fournit un ensemble de lignes directrices et des capacités d'infrastructure générales nécessaires à l'application des normes d'interface EMS-API. Cette partie de la CEI 61970 décrit des scénarios d'intégration types pour l'application de ces normes et l'intégration des types d'applications. Un modèle de référence est défini pour fournir un cadre général d'application des autres parties des normes EMS-API. Le modèle de référence est fondé sur une architecture à composants de sorte que les normes portent sur les interfaces des composants pour l'échange d'informations entre des applications dans un environnement de centre de conduite. Alors que l'EMS-API a pour principal objet de prendre en charge l'intégration d'applications au sein du centre de conduite, le modèle de référence est également applicable aux échanges d'informations entre des applications et des systèmes externes à l'environnement du centre de conduite, tels que d'autres centres de commande, des ERA, des ORT et des systèmes de distribution. La présente norme décrit l'objet des autres parties de la norme, y compris le modèle d'information commun (CIM) dans la série CEI 61970-3xx, les spécifications d'interface de composants (CIS) dans la série CEI 61970-4xx et la mise en correspondance des technologies dans la série CEI 61970-5xx.

La présente partie de la série CEI 61970 couvre également des capacités générales requises par l'infrastructure d'intégration pour faciliter l'échange d'informations par l'intermédiaire des interfaces de composants spécifiées par les CIS. Bien que l'infrastructure d'intégration ne fasse pas en soi partie de cette norme, elle est destinée à fournir certains services essentiels à l'appui des normes d'interface EMS-API. Ces services sont énumérés à l'Article 6.

La présente partie de la série CEI 61970 ne spécifie aucune mise en oeuvre individuelle ni aucun produit particulier; elle n'impose aucune restriction en matière de représentation des informations dans une application informatique. La présente norme spécifie les interfaces visibles de l'extérieur (y compris la sémantique et la syntaxe) nécessaires à la prise en charge de l'interopérabilité des produits fournis par différents fournisseurs.

2 Références normatives

Les documents de référence suivants sont indispensables pour l'application du présent document. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

CEI 61970-2, *Energy management system application program interface (EMS-API) – Part 2: Glossary* (disponible en anglais seulement)

CEI 61970-301, *Interface de programmation d'application pour système de gestion d'énergie (EMS-API) – Partie 301: base de modèle d'information commun (CIM)*

3 Termes et définitions

Pour le besoins du présent document, les termes et définitions donnés dans la CEI 61970-2 s'appliquent.

ENERGY MANAGEMENT SYSTEM APPLICATION PROGRAM INTERFACE (EMS-API) –

Part 1: Guidelines and general requirements

1 Scope

This part of the IEC 61970 series provides a set of guidelines and general infrastructure capabilities required for the application of the EMS-API interface standards. This part of the IEC 61970 series describes typical integration scenarios where these standards are to be applied and the types of applications to be integrated. A reference model is defined to provide a framework for the application of the other parts of these EMS-API standards. This reference model is based on a component architecture that places the focus of the standards on component interfaces for information exchange between applications in a control center environment. While the primary objective of the EMS-API is to support the integration of applications within the control center, the reference model is also applicable to information exchanges between control center applications and systems external to the control center environment, such as other control centers, ISOs, RTOs, and distribution systems. This standard describes the role of the other parts of the standard, including the Common Information Model (CIM) in the IEC 61970-3XX series, the Component Interface Specifications (CIS) in the IEC 61970-4XX series, and Technology Mappings in the IEC 61970-5XX series.

This part of the IEC 61970 series also includes general capabilities that are needed by the integration infrastructure to facilitate the exchange of information via the component interfaces specified by the CIS. While the integration infrastructure itself is not part of this standard, it is expected to provide certain essential services to support the EMS-API interface standards. These services are enumerated in Clause 6.

This part of the IEC 61970 series does not specify individual implementations or products, nor does it constrain the representation of information within a computer system application. This standard specifies the externally visible interfaces, including semantics and syntax, required to support the interoperability of products supplied by different vendors.

2 Normative References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 61970-2, *Energy management system application program interface (EMS-API) – Part 2: Glossary*

IEC 61970-301, *Energy management system application program interface (EMS-API) – Part 301: Common Information Model (CIM) base*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 61970-2 apply.

4 Intégration du système

4.1 Scénarios d'intégration

Les systèmes de gestion d'énergie sont un assemblage de divers sous-systèmes logiciels (SCADA, commande de production, prévision de charge, etc.). Les lignes directrices élaborées dans ce domaine (ou dans une partie de ce domaine) s'appliquent à plusieurs scénarios d'intégration. Il est admis que les systèmes existants à interface unique devront évoluer pour pouvoir utiliser les interfaces spécifiées dans les présentes. Différentes situations d'intégration sont prises en compte; bien qu'il s'agisse de scénarios types, les exemples suivants ne constituent que des sous-ensemble possibles.

a) Intégration d'applications développées par différents fournisseurs dans un système homogène:

   Dans ce scénario, des composants d'application développés de manière indépendante (tels que Optimal Power Flow (OPF)) sont intégrés dans un système (y compris la prise en charge de l'infrastructure). L'intégrateur du système peut ainsi intégrer plus facilement un composant individuel dans un système, qu'il s'agisse d'un système existant ou récemment développé.

b) Echange de données en ligne entre des systèmes distincts:

   Le centre de conduite doit communiquer avec d'autres systèmes distincts au sein de la société et avec d'autres entreprises. L'échange d'informations implique l'utilisation de schémas d'intégration à couplage lâche, par exemple avec un SGD (système de gestion de la distribution), avec le système comptable de la société ou d'autres EMS. Les scénarios d'intégration des entreprises appartiennent généralement à cette catégorie.

c) Intégration de systèmes distincts partageant des données techniques:

   Dans cette situation, des progiciels d'application provenant de différents fournisseurs utilisent des données de modélisation techniques se chevauchant partiellement (telles que l'impédance d'un segment de ligne).

d) Echange de données mises en série entre des applications identiques dans des systèmes différents:

   Il est possible d'obtenir une intégration limitée en utilisant des techniques de transfert de fichiers. Ce type d'échange exportation/importation suppose qu'un format d'échange a été convenu. L'utilisation du protocole de transfert de fichier (FTP) et un fichier (ou document) dont le format est basé sur le langage de balisage extensible (XML) pour échanger des modèles réseau constitue un exemple de ce scénario.

e) Développer une nouvelle application dans un système homogène:

   Cela correspond aux fournisseurs ou aux entreprises de service public développant de nouvelles applications destinées à être intégrées dans leurs systèmes existants (et non dans d'autres systèmes) en utilisant les présentes normes pour l'interface d'application.

4.2 Considérations sur l'intégration

4.2.1 Généralités

Le domaine d'application de l'intégration couvert par les présentes normes est divisé en deux catégories vaguement définies:

a) L'intégration de composants logiciels pour la mise en oeuvre d'un EMS ou d'un système similaire.

b) L'intégration de systèmes indépendants.

4 System integration

4.1 Integration scenarios

Energy management systems are an assemblage of various software subsystems (SCADA, generation control, load forecast, etc.). The guidelines created for this area (or portions thereof) apply to several different integration scenarios. This work recognizes that existing systems with their unique interfaces will need to evolve to use the interfaces specified in this work. The following different types of integration situations are envisaged. While these are typical scenarios, they are only a subset of the possible examples.

a) Integration of applications developed by different suppliers into one homogenous system:

   In this scenario, independently developed application components (such as an Optimal Power Flow (OPF)) are integrated into a system (which includes supporting infrastructure). This allows the system integrator to more easily integrate an individual component into any system, whether it is an existing system or a newly developed system.

b) On-line data exchange between separate systems:

   The control center needs to communicate with other, separate systems within the corporation and between enterprises. Loosely coupled integration schemes are needed to exchange information. Examples include information exchanges with a DMS, the corporate accounting system, or another EMS. Enterprise integration scenarios generally fit into this category.

c) Integration of separate systems sharing some engineering data:

   This corresponds to the situation where application packages from different vendors use partly overlapping engineering modeling data (such as the impedance of a line segment).

d) Exchange of serialized data between the same applications in different systems:

   Limited integration can be achieved by using file transfer techniques. An agreed upon format is used for this type of export/import exchange. The use of File Transport Protocol (FTP) and an eXtensible Markup Language (XML)-based format file (or document) for the exchange of power sysem models is an example of this scenario.

e) Developing a new application in a homogenous system:

   This corresponds to vendors or utilities developing new applications for integration into their existing systems (as opposed to integration in other systems) using these standards for the application interface.

4.2 Integration considerations

4.2.1 General

The scope of integration to be supported by these standards falls into two loosely defined categories:

a) Integration of software components to implement an EMS or similar system.

b) Integration of independent systems.

Pour assurer une interaction efficace entre les composants logiciels et les systèmes, il est nécessaire d'utiliser un modèle d'information commun (CIM) qui attribue une signification commune et cohérente (c'est-à-dire une sémantique) aux informations échangées ou consultées. Par exemple, pour échanger des informations sur l'impédance d'un transformateur, la classification de l'équipement en tant que transformateur doit être fournie avec le nom de l'attribut pour la valeur de l'impédance. A l'aide de ces informations, il est possible de déterminer l'instance précise d'un transformateur selon l'impédance correspondante. Le nom sémantique de ces objets du monde réel (accompagnés de leurs attributs, descriptions et relations avec d'autres objets du monde réel) est fourni par le modèle CIM (voir 6.2).

Toutefois, les deux catégories d'intégration traitées dans les articles suivants diffèrent légèrement sur d'autres aspects. La collection de normes EMS-API vise à traiter les deux catégories, celles-ci étant considérées comme présentant des interfaces de composants.

4.2.2 Intégration de composants logiciels dans un système

4.2.2.1 Généralités

Plusieurs problèmes doivent être résolus avant d'intégrer des composants logiciels développés de manière indépendante dans un système.

a) Interactions des composants logiciels:

   L'interaction des composants logiciels doit être collaborative pour que le système fonctionne correctement. L'interaction peut se dérouler sous la forme d'accès à une propriété, d'invocation de méthode ou de traitement des événements. Les interfaces publiques qui sont constituées de propriétés, de méthodes et d'événements doivent être identifiées et un contrat sur leur utilisation doit être spécifié pour prendre en charge l'intégration des composants logiciels. Lorsqu'il existe des scénarios d'interaction similaires au sein d'un système, la spécification de formes d'interface cohérentes facilite l'intégration et la maintenance du système.

b) Données de modèles techniques publiques pour l'initialisation:

   Le système électrique physique (et les informations associées) est simulé avec des données techniques comme indiqué dans le modèle CIM. Les composants logiciels partagent des aspects de ces données techniques pour exécuter leurs fonctions. Avant de lancer le composant logiciel, un composant doit être initialisé avec un modèle cohérent et exact du système de monde réel qu'il simule. Une interface commune d'accès aux données publiques partagées constitue un mécanisme cohérent pour que les composants logiciels initialisent leurs modèles internes. Après l'initialisation des composants logiciels, les mécanismes d'interaction peuvent être utilisés pour la mise à jour des modèles techniques.

c) Paquetages pour déploiement:

   Les composants logiciels doivent être réalisés dans des formes spécifiques de paquetage et de livraison à l'intégration du système. Il existe quelques cadres technologiques principaux dans l'industrie du logiciel actuelle, chacun ayant son propre format de paquetage. Il convient que la spécification des normes favorise les possibilités de mise en oeuvre en facilitant l'intégration de logiciels développés de manière indépendante. Pour atteindre ce but, il convient de spécifier les interactions entre les composants logiciels sous une forme abstraite capable de s'adapter à des spécialisations particulières de technologie et de langage.

Pour prendre en charge ce type de scénario d'intégration, les propriétés, les méthodes et les événements utilisés au niveau de l'interface de composants doivent être normalisés ainsi que le contenu des données des informations échangées.

To properly interact, both software components and systems need a Common Information Model (CIM) to provide a common, consistent meaning (i.e., semantics) to the information exchanged or accessed. For example, to exchange information about a transformer impedance, the classification of the piece of equipment as a transformer is needed together with the attribute name for the impedance value. With this information, a particular instance of a transformer can be identified with its corresponding impedance. The semantic name of these real world objects (together with their attributes, descriptions, and relationships to other real world objects) is provided by the CIM (see 6.2).

However, in other respects, these two categories of integration as discussed in the following clauses have slightly different needs. The intent of the EMS-API series of standards is to address both, and both are viewed as presenting component interfaces.

4.2.2 Integration of software components into a system

4.2.2.1 General

To integrate independently developed software components into a system, several issues need to be addressed.

a) Software component interactions:

   Software components need to interact in a collaborative fashion for the system to function properly. This interaction can be in the form of property access, method invocation, and event handling. The public interfaces consisting of properties, methods, and events shall be identified, and a contract on their use shall be specified in order to support integration of software components. Where similar interaction scenarios exist within the system, the specification of consistent interface patterns simplify system integration and maintenance.

b) Public engineering model data for initialization:

   The physical power system and related information is simulated with engineering data as reflected in the CIM. Software components share aspects of this engineering data to perform their functions. Upon software component start-up, a component needs to be initialized with a consistent, accurate model of the real-world system it is simulating. A common interface for access to this public, shared data provides a consistent mechanism for software components to initialize their internal models. Once initialized, software component interaction mechanisms can be used to keep their engineering models up to date.

c) Packaging for deployment:

   Software components need to be realized in specific forms for packaging and delivery to system integration. A few major technology frameworks exist in the software industry today, each with their own format for packaging. The specification of standards should promote flexibility in implementation possibilities while facilitating the integration of independently developed software. To accomplish this, software component interactions should be specified in an abstract form that is capable of particular technology and language specializations.

To support this type of integration scenario, the properties, methods, and events used at the component interface need to be standardized as well as the data content of the information exchange.

4.2.2.2 Catégories d'application

Le domaine d'application des normes EMS-API couvre toutes les applications figurant généralement dans un environnement de centre de conduite ainsi que les interfaces avec des systèmes externes nécessaires pour prendre en charge des opérations en temps réel. Dans la mesure où la norme EMS-API vise davantage à définir des *normes d'interface* que des applications standard, il est plus aisé de comprendre le domaine d'application du projet en analysant la liste des catégories d'applications couvertes par les normes EMS-API. Dans la mesure où le paquetage réel en applications des interfaces de composants spécifiées dans les CIS relève des fournisseurs d'applications, toute tentative d'établir une liste d'applications particulières en mentionnant leur nom représenterait une contrainte inutile pour les fournisseurs.

La liste des catégories d'application et des applications types couvertes par la présente norme figure dans le Tableau B.1. Un résumé de cette liste est fourni ci-dessous:

- système d'acquisition et de contrôle des données (SCADA);
- traitement des alarmes;
- traitement de topologie;
- applications réseau;
- gestion des charges;
- commande de production;
- prévision de charge;
- planification de l'énergie/du transport;
- apurement comptable;
- planification de la maintenance;
- archives d'informations historiques;
- gestion des données;
- interface utilisateur générique;
- simulation dynamique;
- simulateur de formation du répartiteur;
- systèmes externes (par exemple, système de gestion de la distribution (SGD), conditions atmosphériques, vente d'énergie électrique de gros, etc.).

4.2.3 Intégration de systèmes

Pour ce qui concerne l'intégration de systèmes hétérogènes susceptibles de ne présenter aucune similarité dans l'environnement d'exploitation ni de maîtrise de cet environnement, l'accent est davantage mis sur la prise en charge des échanges de "documents " à couplage lâche et asynchrones. Dans ce contexte, un “document” peut être considéré comme une structure de données vaste et riche, tel qu'un document XML. L'échange de documents implique que les données échangées sont complexes, structurées et auto-descriptives. Il est plus probable que l'échange implique des transferts d'informations individuelles et atomiques où tous les renseignements relatifs à la manière de traiter les informations et/ou l'action requise pour le transfert sont auto-contenus dans le fichier, que des transactions multi-étapes où le traitement du transfert des informations pourrait dépendre des transferts d'informations ou événements précédents.

4.2.2.2 Application categories

The scope of the EMS-API standards includes all applications typically found within a control center environment as well as interfaces to external systems needed to support real-time operations. However, since the intent of the EMS-API standard is to define interface standards rather than to define standard applications, the scope of the project can best be understood by considering the list of application categories that will be supported by the EMS-API standards. The actual packaging of component interfaces specified in the CIS into applications is left to the application suppliers; therefore, any attempt to define a list of individual applications by name would unnecessarily constrain suppliers.

The list of application categories and typical applications supported by this standard can be found in Table B.1. The following is a summary of the list:

- supervisory Control and Data Acquisition (SCADA);
- alarm processing;
- topology processing;
- network applications;
- load management;
- generation control;
- load forecast;
- energy/transmission scheduling;
- accounting settlements;
- maintenance scheduling;
- historical information archival;
- data engineering;
- generic user interface;
- dynamic simulation;
- dispatcher training simulator;
- external systems (e.g., DMS, weather, wholesale power marketing, etc.).

4.2.3 Integration of systems

In the integration of heterogeneous systems, where there is likely to be no similarity in the operating environment nor control over it, the emphasis is more on supporting loosely coupled, asynchronous "document" exchanges. In this context, a "document" may be thought of as a large, rich data structure, such as an XML document. Document exchange implies that the data exchanged is complex, structured, and self-describing. The exchange is more likely to involve individual, atomic information transfers where all information regarding how to handle the information and/or action requested in the transfer is self-contained, rather than multi-step transactions where the handling of the information transfer may be contingent upon previous information transfers or events.

4.3 Interfaces basées sur des composants

Les normes EMS-API ont notamment pour objet de favoriser le développement indépendant de composants logiciels réutilisables et de faciliter leur intégration dans la construction de systèmes de centre de conduite en élaborant des normes relatives à l'interface de composants. L'industrie du logiciel, y compris les principaux fournisseurs de Systèmes de Gestion d'Energie (EMS) et les fournisseurs de logiciels d'application pour EMS ont évolué de l'application de concepts de génie logiciel fondés sur une conception logicielle modulaire descendante en passant par des approches orientées objet pour parvenir au dernier raffinement en la matière, à savoir l'utilisation d'architectures basées sur les composants. Cette tendance est bien reflétée par les modèles de composants adoptés par CORBA®[4] (Common Object Request Broker Architecture), EJB® (Enterprise Java Beans)® de Sun®[5] et DCOM (Distributed Common Object Modeling) de Microsoft®[6]. (Voir l'Annexe A pour une description de ces trois modèles de composants).

Les approches basées sur les composants facilitent également l'intégration de logiciels et de systèmes complets provenant de sources diverses. Pour ce type d'intégration sans contrainte, les services Web XML fournissent un autre modèle d'intégration basé sur Internet. Les services Web XML permettent aux applications de communiquer et de partager des données sur Internet en utilisant un type d'échange d'informations auparavant désigné par "échange de documents", quel que soit le système d'exploitation ou le langage de programmation. Ces services constituent un autre exemple d'environnement d'exécution de composants qui est de plus en plus répandu dans les échanges d'informations du commerce électronique interentreprises (B2B). (Voir annexe A pour une description des services Web XML).

En matière d'EMS-API, cela a pour conséquence d'orienter les travaux davantage sur le développement de normes d'interfaces de composants logiciels pour échanger et accéder à des informations publiques plutôt que sur la normalisation des services du cadre d'intégration fournissant ces capacités. Le but est que les applications conformes à ces normes puissent par la suite être fournies individuellement et réutilisées dans de multiples systèmes. Bien que d'autres services d'infrastructure puissent se révéler nécessaires dans un système réel, tels que des services d'annuaire et de sécurité, ils ne font pas l'objet de la présente norme. Ces services peuvent être considérés effectivement, et à juste titre, comme relevant du domaine de l'intégrateur du système ou du fournisseur du système (c'est-à-dire comme faisant partie de la plate-forme du système EMS et non comme un composant réutilisable prêt à l'emploi).

L'objectif n'est pas de développer des interfaces standard pour les intergiciels mais, au contraire, de garantir l'indépendance vis à vis de tout ensemble particulier de services intergiciels. Cela permet aux intégrateurs de sélectionner correctement le type et l'échelle de l'infrastructure requise par chaque système et aux concepteurs de services d'évoluer et d'innover tout en simplifiant également le développement des composants.

Cela signifie également qu'un développeur de logiciels n'est pas tenu de traiter directement avec ces services précis. L'intégrateur du système fournit la "colle" qui permet d'insérer ces composants dans l'environnement du système. Il dispose d'une plus grande liberté de configuration des composants et de choix des services les mieux adaptés aux besoins du système en cours de mise en oeuvre (c'est-à-dire la performance, la disponibilité etc.).

---

4 CORBA est l'appellation commerciale d'un produit distribué par OMG. Cette information est donnée à l'intention des utilisateurs de la présente Norme internationale et ne signifie nullement que la CEI approuve ou recommande l'emploi exclusif du produit ainsi désigné. Des produits équivalents peuvent être utilisés s'il est démontré qu'ils conduisent aux mêmes résultats.

5 Sun est une forme abrégée de Sun Microsystems, une compagnie basée aux Etats-Unis. Cette information est donnée à l'intention des utilisateurs de la présente Norme internationale et ne signifie nullement que la CEI approuve ou recommande cette compagnie.

6 Microsoft est une forme abrégée de Microsoft Corporation, une compagnie basée aux Etats-Unis. Cette information est donnée à l'intention des utilisateurs de la présente Norme internationale et ne signifie nullement que la CEI approuve ou recommande cette compagnie.

4.3 Component-based interfaces

One goal of the EMS-API standards is to encourage the independent development of reusable software components and facilitate their integration in the construction of control center systems through the development of component interface standards. The software industry, including the major Energy Management System (EMS) vendors and suppliers of application software for an EMS, has undergone an evolution from basing software engineering concepts on top down modular software design to object-oriented approaches to the latest refinement using component-based architectures. The component models embraced by the Common Object Request Broker Architecture (CORBA®)[4], Sun®[5]'s Enterprise Java Beans® (EJB®), and Microsoft®[6]'s Distributed Common Object Modeling (DCOM) best exemplify this trend. (See Annex A for a description of these three component models).

These component-based approaches also facilitate the integration of software and complete systems from various sources. For this type of any-to-any integration, XML Web Services provide another Internet-based integration model. XML Web services allow applications to communicate and share data over the Internet using the type of information exchange earlier referred to as "document exchange," regardless of operating system or programming language. These services provide another example of a component execution environment becoming more prevalent in Business-to-Business (B2B) information exchanges. (See Annex A for a description of XML Web Services).

The effect on the EMS-API is to shift the focus to developing standards for software component interfaces for exchanging and accessing public information rather than on standardizing the integration framework services that provide these capabilities. The expectation is that applications that adhere to these standards can then be individually delivered and reused in multiple systems. While other infrastructure services may be needed in an actual system, such as directory services and security, these are not the goal of this standard. In fact, they may be more properly considered the domain of the system integrator or system supplier (i.e., part of the EMS system platform, not a reusable plug-and-play component).

The goal is not to develop standard interfaces for middleware. In fact, the goal is just the opposite – to be independent of any particular set of middleware services. This allows integrators to select the right type and scale of infrastructure for each system. It allows service designs to evolve and innovate, and it simplifies component development as well.

It also means that a software developer does not have to deal directly with these services. The system integrator provides the "glue" to insert these components into the system environment. This gives the integrator more freedom to configure components and choose the services best suited to the needs of the system being implemented (i.e., performance, availability, etc.).

---

4 CORBA is the trade name of a product supplied by OMG. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

5 Sun is shorthand for Sun Microsystems, a USA-based corporation. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the company named.

6 Microsoft is shorthand for Microsoft Corporation, a USA-based corporation. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the company named.

Les deux exemples suivants illustrent l'indépendance des composants:

– L'utilisation de composants CORBA n'implique pas spécifiquement d'utiliser CORBA Notification (ni un système précis) comme système d'événement.
– Les composants COM+ sont écrits exactement de la même manière, qu'ils soient déployés dans un environnement avec ou sans DTC (coordinateur de transaction distribuée) et/ou MSMQ (système de mise en file d'attente et de routage des messages de Microsoft).

Le Tableau 1 énumère certains avantages de l'approche basée sur les composants adoptée par les normes d'interface.

Tableau 1 – Avantages des interfaces basées sur les composants

Répondent explicitement à l'objectif "prêt à l'emploi" des logiciels du projet EMS-API
Ne prescrivent aucune conception globale de système ni un choix et une conception de services particulière
S'alignent sur l'orientation globale de l'industrie du logiciel qui permet l'utilisation d'outils à grande diffusion pour développer des composants et configurer des systèmes
Exemptent le projet EMS-API de trouver ou d'inventer des solutions pour résoudre les nombreux problèmes posés par les logiciels prêts à l'emploi.
Apportent une solution plus complète y compris dans des domaines importants tels que le paquetage (ce qui est enregistré sur le CD du composant), la documentation, le contrôle des versions, etc.
Ne nécessitent pas de conception et/ou prescription de services intergiciels, en particulier des services d'événements, de dénomination ou de transaction; possibilité d'utiliser des produits disponibles dans le commerce pour fournir ces services.
Fournissent une forme canonique aux normes spécifiques à l'entreprise de service public élaborées sur le projet, ce qui permet aux développeurs d'exécuter directement une traduction automatique des définitions d'interface dans leur langage d'interface privilégié: Object Management Group (OMG) / International Standards Organization (ISO), Langage de Définition d'Interface (IDL), syntaxe de l'interface Java ® [7] ou IDL de Microsoft
Permettent au projet EMS-API de se concentrer sur la conception d'interfaces et d'événements pour des applications d'entreprises de service public sans attendre que tous les problèmes liés aux intergiciels soient résolus. Cela permet aux fournisseurs d'entrer plus rapidement sur le marché en proposant des applicatifs dont les interfaces sont conformes à l'EMS-API.

4.4 Relations avec les normes de la série CEI 61968

La série de normes CEI 61968 concerne les interfaces systèmes pour les systèmes de gestion de la distribution. Il existe de nombreuses similarités entre ces normes et celles de la série CEI 61970, non seulement du fait que les domaines d'application se chevauchent mais aussi parce que les normes de la série CEI 61968 se fondent également sur le modèle CIM. La série CEI 61968 s'appuient autant que possible sur la base CIM contenue dans la CEI 61970-301 en l'élargissant pour y inclure des spécialisations supplémentaires de classes existantes et également en ajoutant de nouveaux ensembles de classes à des objets modèles figurant dans la problématique de la distribution. C'est la raison pour laquelle il est nécessaire d'analyser les deux séries de normes, la CEI 61970 et la CEI 61968, qui traitent du modèle CIM pour appréhender le domaine d'application du modèle CIM dans son intégralité.

Les normes de la série CEI 61968 ont également pour objet la définition des échanges d'informations standard entre des fonctions de gestion de la distribution, à l'instar des normes CEI 61970-4xx; cependant, mais elles ne visent pas à définir des interfaces de programmation d'application (c'est-à-dire les services à mettre en oeuvre par les composants). La série CEI 61968 prévoit cependant la possibilité de transférer les messages standard, tels qu'ils y sont définis, aux API définies dans les normes de la série CEI 61970.

---

7 Java est l'appellation commerciale d'un produit distribué par Sun Microsystems. Cette information est donnée à l'intention des utilisateurs de la présente Norme internationale et ne signifie nullement que la CEI approuve ou recommande l'emploi exclusif du produit ainsi désigné. Des produits équivalents peuvent être utilisés s'il est démontré qu'ils conduisent aux mêmes résultats.

The following two examples illustrate this independence of components:

– CORBA components specifically do not require the use of CORBA Notification (or any particular service) as its event system.
– COM+ components are written exactly the same way whether they are deployed in an environment with or without Distributed Transaction Coordinator (DTC) and/or a Microsoft Message Queue (MSMQ).

Table 1 lists some of the benefits of this component-based approach to interface standards.

Table 1 – Benefits of Component-based Interfaces

Explicitly addresses the EMS-API project software plug-and-play goal
Does not prescribe an overall system design nor the choice and design of services
Aligns with overall software industry direction to permit the use of mainstream tools to develop components and configure systems
Frees the EMS-API project from rediscovering and reinventing solutions for the many problems of "plug-and-play" software
Provides a more complete solution including important areas such as packaging (what goes on the component disc CD), documentation, versioning, etc.
Does not require designing and/or prescribing middleware services, particularly event, naming and transaction services, but rather permits the use of commercially-available products to provide these services
Provides a canonical form for the utility specific standards developed on the project, enabling developers to directly machine-translate interface definitions into their preferred interface language: Object Management Group (OMG)/International Standards Organization (ISO) Interface Definition Language (IDL), Java®[7] interface syntax or Microsoft IDL
Allows the EMS-API project to focus on designing interfaces and events for utility applications without having to wait for all the middleware problems to be solved. This permits vendors to get to market quicker with EMS-API-compliant interfaces in their application products

4.4 Relationship to IEC 61968 series of standards

The IEC 61968 series of standards deals with system interfaces for distribution management systems. There is a great deal of similarity between these standards and those contained in the IEC 61970 series of standards, not only because of some overlap in scope, but because the IEC 61968 series of standards are also based on the CIM. The IEC 61968 series of standards build on the CIM Base specified in IEC 61970-301 wherever possible by extending it to include additional specializations of existing classes, but also adding entirely new sets of classes to model objects found in the distribution problem domain. Therefore, to comprehend the entire scope of the CIM, it is necessary to review both the IEC 61970 and IEC 61968 series of standards which deal with the CIM.

The IEC 61968 series of standards are also concerned with defining standard information exchanges between distribution business functions, similar to the IEC 61970-4XX standards, but do not attempt to define application program interfaces (i.e., services to be implemented by components). However, the IEC 61968 series of standards envision that the standard messages defined therein can be transferred over the APIs defined in these IEC 61970 series of standards.

---

[7] Java is the trade name of a product supplied by Sun Microsystems. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

5 Modèle de référence EMS-API

5.1 Généralités

Le modèle de référence EMS-API est une architecture abstraite qui permet de visualiser l'espace problème traité, fournit un langage pour décrire et examiner des solutions, définit une terminologie et apporte d'autres aides similaires facilitant la compréhension mutuelle du problème que les normes EMS-API se proposent de résoudre.

Le modèle de référence n'est pas en soi une conception et n'est pas destiné à décrire les couches de logiciel; une approche par couche apparaît toutefois implicite et inévitable. Ce modèle a pour principale fonction d'indiquer clairement les aspects de l'espace problème faisant l'objet du domaine des normes EMS-API, ceux étant exclus du domaine du projet EMS-API et de justifier ces choix. Il a également pour objet de mettre en évidence les liens entre les différentes parties de la norme.

La Figure 1 présente un schéma du modèle de référence; les zones ombrées représentent les portions du modèle de référence couvertes par la présente norme. Les zones non ombrées représentent les parties d'un système essentielles pour créer un cadre aux composants d'application réutilisables. Toutes les parties de ce modèle sont traitées dans le présent document.

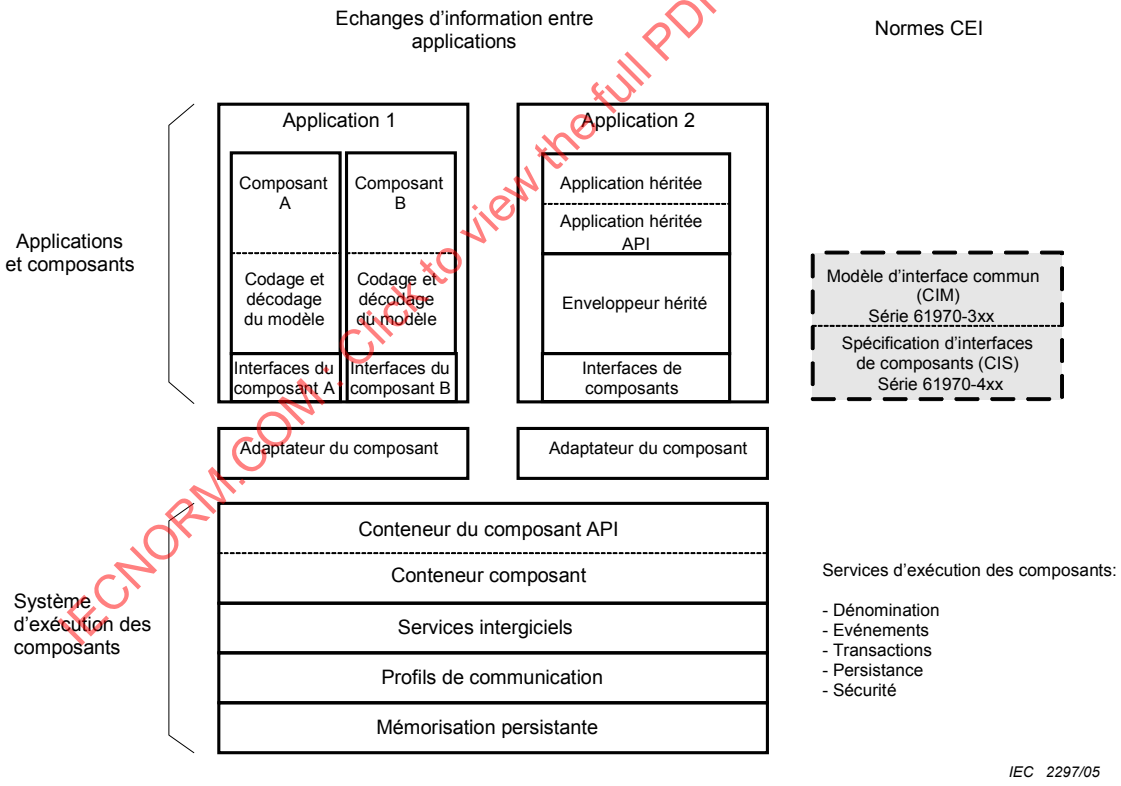

Figure 1 – Modèle de référence EMS-API

5 EMS-API reference model

5.1 General

The EMS-API reference model is an abstract architecture that provides a visualization of the problem space being addressed, provides a language for describing and discussing solutions, defines terminology, and provides other similar aids toward achieving a mutual understanding of the problem being solved with the EMS-API standards.

The reference model is not a design, nor is it intended to describe software layers, although a layering approach is hard to avoid and is implied. The primary function is to show clearly which parts of the problem space are the subject of the EMS-API standards, and by contrast, which are outside the domain of the EMS-API project and why. It is also intended to show how different parts of the standard relate to each other.

Figure 1 is a diagram of the reference model, with the shaded areas representing those portions of the reference model that are the subject of this standard. The non-shaded areas represent parts of a system that are essential for creating a framework for reusable application components. Each part of the model is discussed later in this document.

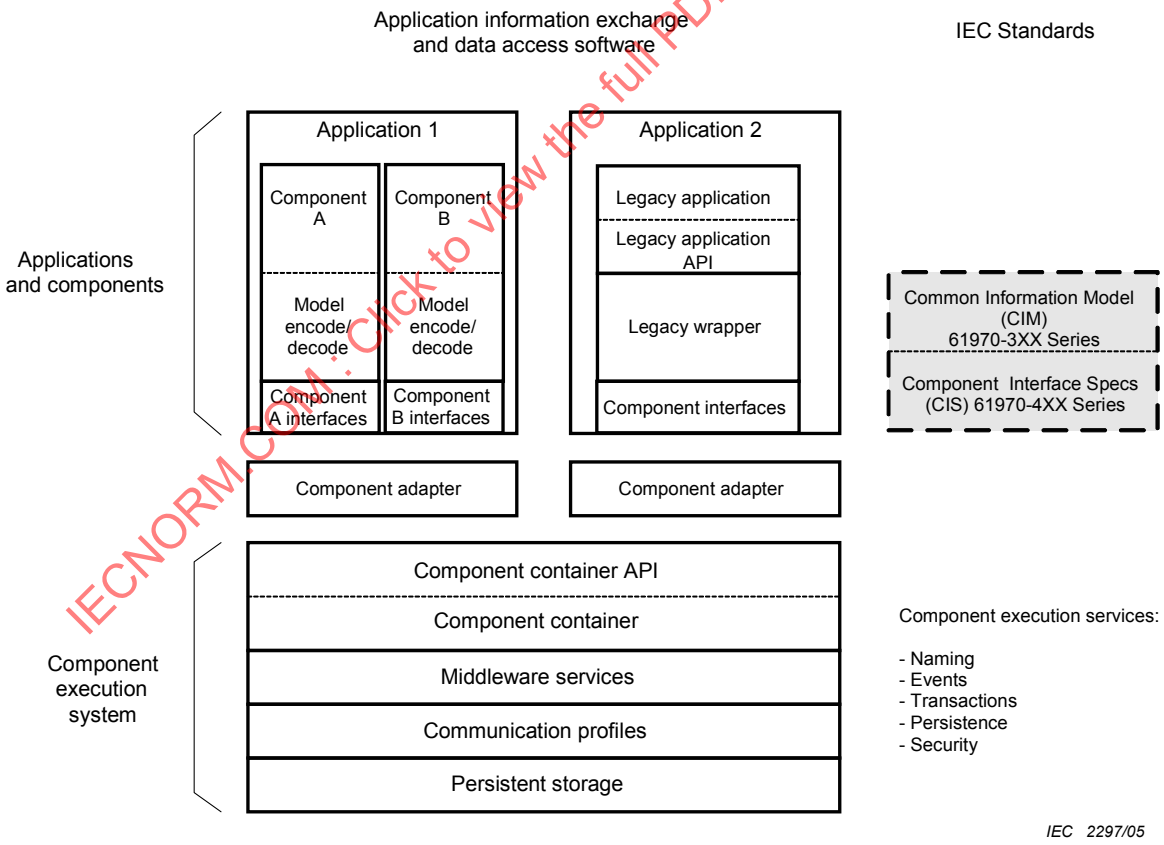

Figure 1 – EMS-API Reference Model

5.2 Environnement du centre de conduite

Le modèle de référence est spécifiquement prévu pour s'appliquer aux environnements de centre de conduite comprenant généralement des réseaux d'ordinateurs connectés en réseau local (LAN) et parfois en réseau étendu (WAN). Un centre de conduite peut comporter plusieurs systèmes supportant des opérations d'une entreprise de service public, y compris un EMS (système de gestion d'énergie), un SGD (système de gestion de la distribution) et d'autres systèmes requis pour les fonctions de gestion des ERA (exploitants de réseau autonomes) et des ORT (organisations régionales de transport). Il peut accueillir plusieurs groupes d'utilisateurs et fonctions organisationnelles, y compris des exploitants de service, des superviseurs, la formation des exploitants, la planification des opérations, la maintenance des bases de données et le développement de logiciels. Dans un EMS, de nombreuses applications sont utilisées dans plusieurs de ces contextes; il est important de pouvoir configurer facilement (de préférence automatiquement) une application pour l'utiliser dans de multiples contextes. Une telle configuration est obtenue en associant des *propriétés* à chaque interface de composants plutôt qu'en modifiant le code interne du composant.

5.3 Contexte d'application

Un contexte d'application comprend un ensemble d'applications qui travaillent ensemble en tant qu'unité organisationnelle pour atteindre des objectifs de haut niveau [8]. Un contexte d'application définit un délai et un environnement d'exécution. Des exemples de contextes d'applications EMS sont fournis dans le Tableau 2.

Tableau 2 – Exemples de contextes d'application EMS

Temps réel	Commande en ligne du système électrique
Etude des opérations	Exécution d'applications réseau pour étudier et/ou analyser des pratiques d'opérations (très court terme)
Etude de la planification de l'extension	Exécution d'études réseau et/ou simulation pour évaluer d'autres solutions possibles (à l'avenir/à long terme)
Formation	Fournit un environnement de formation pour les exploitants, nécessitant la simulation et des applications d'analyse.

Bien que le modèle de référence ne l'indique pas explicitement, plusieurs contextes peuvent évidemment exister simultanément, chacun pouvant impliquer les même applications dans différents échanges de données. Par ailleurs, plusieurs instances d'un contexte particulier peuvent coexister. Par exemple, deux études ou plus peuvent fonctionner en parallèle pour deux exploitants différents dans le contexte "Etude des opérations", pendant que les mêmes applications s'exécutent dans le contexte "Temps réel".

5.4 Application

Une application comprend un ou plusieurs composants réalisant une fonction de gestion dans un domaine donné. Elle est conçue et écrite par un expert du domaine. La granularité des composants comprenant l'application relève du choix du concepteur. Des générateurs d'application peuvent combiner des composants de différents développeurs ou de différents fournisseurs pour construire une application.

---

8 Il convient de noter que le terme *contexte* est utilisé différemment lorsqu'il s'agit de modèles de composants pour EJB et CORBA, par exemple, lorsque le contexte est utilisé en relation avec un conteneur de composants assurant la mise en oeuvre de composants avec accès aux services d'exécution mis en oeuvre par le conteneur. Ces services comprennent les transactions, la sécurité, les événements et la persistance.

5.2 Control center environment

The reference model is specifically intended to apply to control center environments, which typically comprise networks of computers connected by Local Area Networks (LANs) and sometimes Wide Area Networks (WANs). A control center may include a variety of systems in support of utility operations, including EMS, DMS, and other systems needed for ISO and RTO business functions. It supports a number of different user groups and organizational functions, including on-duty operators, supervisors, operator training, operations planning, database maintenance, and software development. In an EMS, many applications are used in a number of these contexts, and it is important that an application can be easily configured (preferably automatically) for use in multiple contexts. This is accomplished through the use of *properties* that are associated with each component interface rather than through changes to the internal code of the component.

5.3 Application context

An application context comprises a collection of applications working together as an organizational unit to accomplish some high-level objective.[8] An application context defines a time frame and an environment for execution. Table 2 includes examples of contexts for EMS applications:

Table 2 – Examples of EMS application contexts

Real Time	On-line control of the power system
Operations Study	Execution of network applications to study and/or analyze operations practices (near-term)
Extension Planning Study	Execution of network and/or simulation studies to evaluate alternatives (future/long term)
Training	Provide training environment for operators, requiring simulation and analysis applications

While not shown explicitly in the reference model, it is understood that several contexts may exist simultaneously, each involving potentially the same applications in different data exchanges. Furthermore, there may be multiple instances of a particular context coexisting. For example, there may be two or more studies running in parallel for two different operators in the Operations Study context, while concurrently the same applications are executing in the Real Time context.

5.4 Application

An application comprises one or more components that perform some business function in a given domain. It is designed and written by a domain expert. The granularity of the components comprising the application is the designer's choice. Application builders can combine components from different developers or different vendors to construct an application.

---

8 It should be noted that the term *context* is used differently in the discussion of component models for EJB and CORBA, for example, where context is used in connection with a component container providing component implementations with access to the runtime services implemented by the container. These services include transactions, security, events, and persistence.

Il convient qu'un développeur d'application puisse utiliser pleinement un composant sans avoir besoin d'accéder à son code source. Les composants peuvent être personnalisés pour répondre aux exigences spécifiques d'une application grâce à un ensemble de valeurs de propriété externes. Par exemple, le composant bouton dispose d'une propriété qui spécifie le nom devant apparaître sur le bouton. Il est évident que le nombre de personnalisations admissibles dépend des valeurs de propriété externes prévues par le développeur de composants. Cela équivaut en quelque sorte à l'ancien concept de conception de programmation permettant de personnaliser un programme en spécifiant des valeurs appropriées pour des paramètres configurables plutôt qu'en modifiant le code source.

Le Tableau B.1 fournit une liste des catégories d'applications, des noms abstraits et des fonctions réalisées.

5.5 Composant

Un composant est un bloc fonctionnel logiciel réutilisable. Il s'agit d'une pièce préfabriquée d'un code d'application encapsulé qui peut être associée à d'autres composants et à un code écrit spécifiquement pour produire rapidement une application personnalisée.

Pour être qualifié de composant, un code d'application doit fournir une interface standard qui permet à d'autres parties de l'application d'appeler ses fonctions, d'accéder à des données à l'intérieur du composant et de les traiter. Le modèle de composants définit la structure de l'interface.

Les composants diffèrent les uns des autres en fonction de leur granularité. Un composant peut être de très petite taille, tel qu'un simple objet d'interface graphique (par exemple un bouton) ou il peut mettre en oeuvre toute une application complexe, telle qu'une application d'estimateur d'état. Dans ce dernier cas, l'application pourrait être conçue à partir de zéro comme un composant unique, ou elle pourrait comprendre une application héritée enveloppée de sorte à être conforme aux normes d'interface de composants (voir ci-dessous pour ce qui concerne les applications héritées).

Un composant peut être installé sur un support transportable, tel qu'un CD, et expédié pour être utilisé dans un système fournissant des conteneurs de composants.

De manière générale, les composants exposent ouvertement des méthodes, des propriétés et des "événements" par l'intermédiaire d'une infrastructure d'intégration. Les événements sont particulièrement intéressants dans la mesure où ils permettent l'intégration de composants développés de manière indépendante. En utilisant un ensemble standard d'événements, le composant A n'a pas besoin de connaître les détails de l'interface avec le composant B ni même de savoir que le composant B existe. L'élément déterminant est que la norme porte sur l'interface avec le composant et non sur l'infrastructure d'intégration.

5.6 Application héritée et enveloppeurs

Une application héritée diffère légèrement de la définition donnée précédemment d'une application. Une application héritée peut être une application unique capable de réaliser des fonctions de gestion qu'une entreprise de service public peut avoir acheté ou développé elle-même avant d'établir tout modèle de composants à des fins d'intégration. Il peut également s'agir d'un système complet servant de source et/ou de puits pour les données requises/publiées par d'autres systèmes qui doivent être intégrées ensemble pour faciliter l'échange d'informations.

Une application héritée est par exemple:

– une application de gestion de production mise en oeuvre dans FORTRAN sans égard aux interfaces de composants;

An application developer should be able to make full use of a component without requiring access to its source code. Components can be customized to suit the specific requirements of an application through a set of external property values. For example, the button component has a property that specifies the name that should appear on the button. Of course, the amount of customization allowable depends on the foresight of the component developer in providing sufficient external property values. This is somewhat equivalent to the older concept in program design of customizing a program through specifying appropriate values for configurable parameters rather than having to modify source code.

Refer to Table B.1 for a list of application categories, abstract names, and functions performed.

5.5 Component

A component is a reusable software building block. It is a pre-built piece of encapsulated application code that can be combined with other components and with handwritten code to rapidly produce a custom application.

In order to qualify as a component, application code shall provide a standard interface that enables other parts of the application to invoke its functions and to access and manipulate the data within the component. The component model defines the structure of the interface.

Components vary in their granularity. A component can be very small, such as a simple Graphic User Interface (GUI) widget (e.g., a button) or it can implement an entire complex application, such as a state estimator application. In the latter case, the application could be designed from scratch as a single component, or it could comprise a legacy application wrapped to conform to component interface standards (see below for a discussion of legacy applications).

A component can be put on a transportable medium, such as a CD, and shipped for use in a system that provides component containers.

Generally, components publicly expose methods, properties, and "events" via an integration infrastructure. Events are of particular interest because they allow the integration of independently-developed components. By using a standard set of events, Component A need not know any other details of the interface to Component B or even if Component B exists. The key point is that it is the interface to the component that is standardized, not the integration infrastructure.

5.6 Legacy application and wrappers

A legacy application is quite different from the definition of an application given earlier. A legacy application can be a single application performing some business function that a utility may have purchased or developed itself prior to establishing any component model for integration purposes, or it could be an entire system that is a source/sink for data needed/published by other systems that needs to be integrated together to facilitate information exchange.

Examples include:

– a Unit Commitment application implemented in FORTRAN without regard for component interfaces;

– une transmission EMS complète sans interfaces ouvertes publiées qui est nécessaire pour fournir des données SCADA aux réseaux de distribution et, inversement, pour recevoir des mises à jour de son modèle de système électrique provenant des systèmes de gestion des pannes.

Un enveloppeur hérité permet d'encapsuler une application/un système hérité non conforme aux normes d'interface de composants. Il convertit les données d'entrée/de sortie d'un programme hérité en une ou plusieurs interfaces de composants de sorte à pouvoir participer à l'échange d'informations qui se déroule dans une architecture basée sur les composants.

L'utilisation d'un enveloppeur hérité permet à une application/un système hérité de fonctionner comme un composant prêt à l'emploi capable d'échanger des informations avec d'autres composants par l'intermédiaire d'une infrastructure commune ou d'un cadre commun.

Les enveloppeurs hérités peuvent être conçus et mis en oeuvre par l'expert du domaine qui a développé ou qui détient l'application/le système hérité (par exemple le fournisseur de l'EMS), qui fournit ensuite l'enveloppeur pour l'utilisation dans plusieurs installations du système utilisant la technologie des composants. Dans le cas où le programme hérité est une application "entreprise" personnalisée et isolée, l'intégrateur du système peut devoir écrire un enveloppeur qui ne sera utilisé que sur un seul système.

5.7 Modèle de composants

Un modèle de composants définit l'architecture de base d'un composant en spécifiant la structure de ses interfaces et les mécanismes d'interaction avec le composant, son conteneur et d'autres composants. Le modèle de composants fournit des lignes directrices pour créer et mettre en oeuvre des composants capables de fonctionner ensemble pour former une plus grande application.

Un générateur de composants ne devrait pas avoir à implémenter des services de traitement multiprocessus («multithreading»), de contrôle d'accès simultané, de mise en commun des ressources, de sécurité, de gestion des transactions dans chaque composant. De surcroît, si ces services étaient mis en oeuvre dans chaque composant, il serait très difficile d'obtenir un ensemble d'applications véritablement prêt à l'emploi. Le modèle de composants normalise et automatise l'utilisation de ces services.

Il existe quatre principaux modèles de composants largement répandus au sein de l'industrie du logiciel actuelle. Ces modèles de composants sont décrits à Annexe A. Comme il est spécifié dans ces quatre modèles de composants, l'industrie du logiciel a estimé que la suppression de la reconnaissance du conteneur ou de l'infrastructure par les composants constitue une avancée considérable vers une ère où les composants pourront être développés, assemblés et déployés de manière indépendante. Cependant, dans la mesure où il existe quatre modèles différents, il est possible que des fournisseurs de composants souhaitent fournir une version légèrement différente de chaque modèle pour faciliter l'utilisation de toutes les caractéristiques disponibles du conteneur et du système d'exécution sous-jacent ainsi que des performances inhérentes qu'il peut comporter. Cependant, cela n'est pas absolument nécessaire. Dans la mesure où les fournisseurs de composants imposent des restrictions à leur conception de composants pour assurer une bonne exploitation sur les quatre modèles, les différences peuvent être compensées par des adaptateurs de composants fournis par l'intégrateur du système qui a choisi la technologie de composants spécifique devant être utilisée pour une mise en oeuvre spécifique du système. Dans d'autres cas, l'intégrateur du système peut choisir de combiner plusieurs technologies de composants et utiliser des technologies de transition pour permettre aux différentes technologies de fonctionner ensemble (par exemple, un système d'exécution CORBA fonctionnant avec un système d'exécution Microsoft DCOM).

– an entire transmission EMS without open, published interfaces that is needed to supply SCADA data to distribution systems and in turn receive updates to its power system model from outage management systems.

A legacy wrapper is used to encapsulate a legacy application or system that does not conform to the component interface standards. It converts a legacy program input/output into one or more component interfaces so it can participate in information exchange in a component-based system architecture.

The purpose of using a legacy wrapper, then, is to permit a legacy application or system to operate as a plug-and-play component capable of exchanging information with other components via a common infrastructure or framework.

Legacy wrappers may be designed and implemented by the domain expert who developed or owns the legacy application or system (e.g., the EMS vendor), who then supplies the wrapper for use in multiple system installations where component technology is used. Alternatively, where the legacy program is a one-of-a-kind custom "enterprise" application, the system integrator may have to write the wrapper and it would end up being used in only one system.

5.7 Component model

A component model defines the basic architecture of a component, specifying the structure of its interfaces and the mechanisms by which it interacts with its container and with other components. The component model provides guidelines to create and implement components that can work together to form a larger application.

A component builder should not have to implement multithreading, concurrency control, resource-pooling, security, and transaction management in every component. Furthermore, if these services were implemented in each component, achieving true plug-and-play application assembly would be very difficult. A component model standardizes and automates the use of these services.

There are four primary component models with wide acceptance within the software industry today. These component models are described in Annex A. The software industry has decided, as specified in all four of these component models, that removing container or infrastructure awareness from components is a major step toward a world where components can be independently developed, assembled and deployed. However, since there *are* four separate models, component vendors may want to provide a slightly different version for each to facilitate the use of all available features of the underlying container and execution system, with the inherent performance advantages that may entail. This is not strictly necessary, however. As long as component vendors constrain their component design to ensure correct operation in any of the four models, the differences can be made up by component adapters supplied by the system integrator who has chosen the specific component technology to be used for a specific system implementation. Alternatively, the system integrator may choose to combine more than one component technology and use bridging technologies to permit interoperation of the different technologies (e.g., CORBA execution system interoperating with a Microsoft DCOM execution system).

5.8 Conteneur de composants

Les composants s'exécutent au sein d'un conteneur. Un conteneur sert de contexte à un ou plusieurs composants et fournit des services de gestion et de commande aux composants. En pratique, dans des systèmes, un conteneur fournit un processus de système d'exploitation ou un fil (thread)) pour le composant. Il isole le composant de la plate-forme d'exécution. Lorsqu'un client appelle un serveur de composant, le conteneur attribue automatiquement une unité d'exécution de processus et lance le composant. Le conteneur gère toutes les ressources pour le compte du composant et toutes les interactions entre le composant et les systèmes externes.

Les conteneurs de composants sont généralement fournis par le fournisseur du système comme faisant partie intégrante du système d'exécution du composant. Les services types fournis pour le compte des composants sont la dénomination, les événements, les transactions, la sécurité et la persistance. Tous les services nécessaires à une application en temps réel d'une entreprise de service public sont susceptibles de ne pas être fournis comme faisant partie intégrante des services conteneur standard proposés par les fournisseurs de logiciels commerciaux. L'Annexe C décrit comment ces services peuvent être fournis dans une mise en oeuvre réelle.

5.9 Adaptateur de composant

Les opérations et les comportements d'un conteneur sont définis et dictés par son modèle de composants. Le modèle de composants fournit un contrat relatif à la fourniture des services et des interfaces du conteneur. En conséquence, les composants développés pour un type de système d'exécution ou d'environnement ne sont en général pas transférables directement à un autre type d'environnement d'exécution. Ainsi, pour réutiliser un composant avec des systèmes d'exécution autres que celui pour lequel il a été conçu, il est nécessaire d'utiliser un adaptateur de composant. Si tel n'est pas le cas, les interfaces de composants peuvent être définies conformément à une norme neutre et un adaptateur de composant peut être nécessaire pour permettre à tous les conteneurs de mettre en correspondance les interfaces standard avec celles fournies par le conteneur. Cela équivaut en quelque sorte à la "couche de transférabilité" faisant partie intégrante de la plate-forme Java Enterprise, le système d'exécution de composants pour EJB.

Un *adaptateur de composant* est défini comme une partie de logiciel située entre l'application (ou le composant), le conteneur de composants et l'infrastructure d'intégration; il fournit des services essentiels de support des composants (par exemple, publier/s'abonner, mise en file d'attente de messages, dénomination, etc.). L'adaptateur peut se charger, si nécessaire, des différences de protocoles, de la transformation et de la traduction de données. Un adaptateur peut également fournir un support de composants pour la sécurité, les transactions et la persistance (du moins, pour ce qui concerne le monde extérieur à l'application) lorsque l'environnement d'exécution du composant ne fournit pas lui-même ces services.

Les adaptateurs de composants concernent tant les différences entre a) l'interface de composants, l'environnement du système d'exécution choisi et la technologie du conteneur et b) une interface de composants et d'autres interfaces de composants utilisées dans le reste du système, y compris les différences de définitions d'événements. Ces différences apparaissent en raison des points suivants:

– Le système est constitué de composants développés de manière indépendante dont les interfaces n'ont pas été coordonnées au préalable (c'est-à-dire en l'absence de conformité à une norme, ce qui est probablement le cas en général).

– Le système comporte des composants partiellement normalisés, mais les parties non normalisées ne sont pas compatibles.

– Bien qu'elles soient normalisées, certaines parties ne sont pas compatibles car elles correspondent à différentes versions de la norme.

5.8 Component container

Components execute within a container. A container provides a context for one or more components and provides management and control services for the components. In practical systems, a container provides an operating system process or thread in which to execute the component. It insulates the component from the runtime platform. When a client invokes a server component, the container automatically allocates a process thread and initiates the component. The container manages all resources on behalf of the component and manages all interactions between the component and the external systems.

Component containers are typically provided by the system supplier as part of the component execution system. Typical services provided on behalf of the components are naming, events, transactions, security, and persistence. All services needed by a utility real-time application may not be provided as part of the standard container services offered by commercial software suppliers. Annex C describes how these services may be provided in an actual implementation.

5.9 Component adapter

The operations and behaviors of a container are defined and dictated by its component model. The component model provides a contract for how the container services and interfaces shall be provided. As a result, components developed for one type of execution system or environment are usually not directly portable to any other type of execution environment. Therefore, in order to achieve reuse with multiple execution systems, a component adapter is required for any execution system except the one for which the component was designed. Alternatively, component interfaces could be defined according to some neutral standard, and then a component adapter would be required for all containers to map the standard interfaces onto those provided by the container. This is somewhat equivalent to the “portability layer” that is part of the Java Enterprise Platform, which is the component execution system for EJB.

A component adapter is defined as a piece of software that sits between the application (or component) and the component container and integration infrastructure, which provides fundamental component support services (e.g., publish/subscribe, message queuing, naming, etc). The adapter can, if needed, take care of protocol differences, data transformation, and data translation. An adapter can additionally provide a component support for security, transactions and persistence (at least from the perspective of the world outside the application) if the component execution environment itself does not provide these.

Component adapters deal with both the differences between a) the component interface and the chosen execution system environment and container technology and b) a component interface and other component interfaces used in the rest of the system. This includes differences in event definitions. These differences arise because:

– The system is made up of independently developed components whose interfaces were not coordinated in advance (i.e., not standardized, which is probably the usual case).

– The system contains partially standardized components, but the non-standard parts are incompatible.

– The standardized parts are still incompatible because they reflect different versions of the standard.

Selon l'environnement et les outils fournis par le conteneur de composants, l'adaptateur de composant peut également connecter les portails d'événement du composant aux domaines d'événement appropriés et localiser les composants de dépendance correspondants, s'ils existent.

Les adaptateurs de composants peuvent également mettre en oeuvre les services spécifiques à l'entreprise de service public nécessaires au fonctionnement des composants dans les contextes EMS définis préalablement et qui ne sont pas disponibles avec les systèmes d'exécution de composants du commerce.

Un intégrateur de système ou un fournisseur de système d'exécution fournit en règle générale des adaptateurs de composants. Ils consistent généralement en des codes écrits 1) pour faire correspondre les types d'événements, les types de données et les services prévus par le composant avec ceux prévus par d'autres composants dans le système particulier mis en œuvre, et 2) pour configurer les flux d'informations du système pour le composant. En conséquence, les intégrateurs de système les personnalisent selon le système et selon le composant.

5.10 Système d'exécution de composants

Les composants d'un serveur s'exécutent dans un environnement fourni par une application ou un système d'exécution de composants. L'expression "système d'exécution de composants" englobe la totalité du modèle de référence depuis la couche conteneur jusqu'à la couche inférieure, y compris le conteneur de composants, les services intergiciels et les profils de communication. Il comprend les autres services fournis en plate-forme normale également cachés, y compris le système d'exploitation, le stockage rémanent, etc. Ils sont également qualifiés de systèmes conteneurs dans la mesure où le conteneur fournit la première interface aux normes d'interface faisant l'objet de la présente norme.

Les systèmes conteneurs comportent des intergiciels existants qui respectent le contrat et les politiques du conteneur prévus pour le modèle de composants adopté par le système. Tout cadre de système d'exécution respectant le contrat de conteneur pour la prise en charge des composants est valable. Par exemple, un fournisseur d'EMS peut concevoir un système d'exécution d'application EMS pour prendre en charge des conteneurs et permettre l'utilisation de composants développés en interne ou achetés à un autre fournisseur de composants. L'Annexe D comporte d'autres exemples commerciaux de ce type.

Les systèmes d'exécution de composants sont généralement donnés par le fournisseur du système.

5.11 Intergiciel

Le terme intergiciel décrit un groupe diversifié de produits logiciels fonctionnant comme une couche d'intégration, de conversion et/ou de traduction. Les intergiciels fournissent des interfaces génériques pour les événements, la messagerie, l'accès aux données, les transactions, etc.

Les fournisseurs d'intergiciels fournissent des produits qui prennent en charge certaines combinaisons de services génériques par l'intermédiaire de leurs propres interfaces propriétaires. En général, ils ne ciblent pas une industrie précise bien qu'ils fournissent souvent des convertisseurs pour des applications largement utilisées, telles que PeopleSoft®[9], SAP, etc. Ils peuvent fournir ou non tous les services nécessaires dans un environnement d'opérations en temps réel d'une entreprise de service public. Chaque entreprise de service public peut choisir des fournisseurs d'intergiciels différents en conséquence d'une décision sur

---

9 PeopleSoft est l'appellation commerciale d'un produit distribué par Oracle. Cette information est donnée à l'intention des utilisateurs de la présente Norme internationale et ne signifie nullement que la CEI approuve ou recommande l'emploi exclusif du produit ainsi désigné. Des produits équivalents peuvent être utilisés s'il est démontré qu'ils conduisent aux mêmes résultats.

Depending on the environment and tools provided by the component container, the component adapter might also connect the component's event portals to the appropriate event topics and locate the correct dependency components, if any.

Component adapters may also implement the utility specific services needed for components to operate in the EMS contexts identified earlier that are not available with off-the-shelf component execution systems.

A system integrator or execution system supplier typically provides component adapters. They typically consist of code written to 1) match the event types and data types and services expected by the component to those expected by other components in the particular system being implemented, and 2) configure the system information flows for the component. As a result, they are custom crafted per system and per component by the system integrators.

5.10 Component execution system

Server components execute in an environment provided by an application or component execution system. The component execution system term encompasses the entire reference model from the container layer down, including the component container, middleware services, and communication profiles. It includes the other normal platform-supplied services not shown as well, including the operating system, persistent storage, etc. These are also referred to as container systems since the container provides the primary interface to the interface standards that are the subject of this standard.

Container systems include existing middleware products that adhere to the container contract and policies for the component model embraced by the system. Any execution system frameworks that adhere to the container contract for supporting components qualify. For example, an EMS supplier could design the EMS application execution system to support containers, thereby permitting the use of components either developed internally or purchased from another component supplier. Other commercial examples are contained in Annex D.

The system supplier typically provides component execution systems.

5.11 Middleware

The term middleware is used to describe a diverse group of software products that function as an integration, conversion, and/or translation layer. Middleware provides generic interfaces for events, messaging, data access, transactions, etc.

Middleware vendors provide products to support some combination of these generic services via their own proprietary interfaces. Generally, they do not target a particular industry, although they often provide converters for widely used applications, such as PeopleSoft®[9], SAP, etc. They may or may not provide all the services needed in a utility real-time operations environment.

---

9 PeopleSoft is the trade name of a product supplied by Oracle. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

les technologies de l'information prise à l'échelle de l'entreprise pour fixer une norme propre à la société et ne pouvant pas être modifiée aisément. Les normes d'interface de composants EMS-API doivent donc être écrites de sorte à pouvoir être déployées avec des produits intergiciels variés.

Les intergiciels sont en constante évolution. Il convient que les normes d'interface de composants EMS-API n'excluent pas l'utilisation des produits disponibles présentant une qualité supérieure, si un effort d'intégration est entrepris. L'Annexe D comporte des exemples commerciaux de ce type.

5.12 Profils de communication

Les profils de communication déterminent les protocoles et les services de protocoles particuliers utilisés pour échanger des informations entre différentes plates-formes serveur d'un système d'exécution de composants. Des produits intergiciels fonctionnent sur des profils standard, tels que CORBA et EJB qui utilisent Internet Inter-ORB Protocol (IIOP) sur Transport Control Protocol/Internet Protocol (TCP/IP) pour un fonctionnement soit sur internet soit sur des intranets privés. D'autres utilisent des protocoles et des services propriétaires.

Il est prévu que les spécifications d'interface EMS-API ne soient pas associées à un profil particulier de communication.

5.13 Exemples de modèles de référence

Deux exemples sont fournis pour illustrer l'application des concepts de modèle de référence aux systèmes réels.

5.13.1 Exemple d'un EMS généré conformément aux normes EMS-API

La Figure 2 illustre un système EMS conforme au modèle de référence EMS-API. Les composants d'application individuels sont interconnectés par l'intermédiaire d'un système d'exécution de composants et des adaptateurs de composants qui fournissent les services d'infrastructure nécessaires aux composants pour se reconnaître et communiquer entre eux et avec les mémoires de données publiques dans les différents contextes EMS. Dans le cadre de cet exemple, on suppose l'existence d'un système d'exécution de composants CORBA pouvant directement prendre en charge les composants CORBA. Les applications conformes au modèle de composants CORBA ne nécessitent de ce fait aucun adaptateur de composant. Les données SCADA opérationnelles en temps réel proviennent 1) d'un système hérité SCADA enveloppé dans un enveloppeur hérité pour fournir les interfaces de composants nécessaires et 2) d'un serveur de données Inter-Control Center Communication Protocol (ICCP) connecté à d'autres centres de commande ou postes. Les deux sources de données SCADA publient des mises à jour en utilisant des interfaces de composants identiques lorsqu'elles sont disponibles; les applications nécessitant des mises à jour en temps réel s'abonnent pour les recevoir quelle que soit la source des données. Un système de gestion de la distribution (SGD) externe au centre de conduite est également représenté et qui en s'abonnant peut recevoir les mêmes mises à jour de données SCADA lorsqu'elles sont disponibles. Plusieurs autres interactions entre les applications sont également prises en charge.

Each utility may choose a different middleware vendor as a result of an enterprise-wide Information Technology (IT) decision to set a corporate standard, which may not be easily changed. Therefore, the EMS-API component interface standards shall be written so that they can be deployed with a variety of middleware products.

Middleware products are constantly evolving. The EMS-API component interface standards should not preclude the use of the best available products when an integration effort is undertaken. Annex D provides some examples of commercial products.

5.12 Communication profiles

Communication profiles specify the particular protocols and protocol services that are to be used for information exchange between separate server platforms in a component execution system. Some middleware products operate over standard profiles, such as CORBA and EJB that use Internet Inter-ORB Protocol (IIOP) over Transport Control Protocol/Internet Protocol (TCP/IP) for operation over either the Internet or private intranets. Others use proprietary protocols and services.

The intent of the EMS-API interface specifications is to be independent of any particular communication profile.

5.13 Reference model examples

Two examples are provided to illustrate the application of the reference model concepts to actual systems.

5.13.1 An example of an EMS built with EMS-API standards

Figure 2 illustrates an EMS system that conforms to the EMS-API reference model. Individual application components are interconnected via a component execution system and component adapters that provide the infrastructure services needed by the components to discover and communicate with each other and with the public data stores in the various EMS contexts. For this example, a CORBA component execution system that directly supports CORBA components is assumed. Applications that conform to the CORBA component model then do not need component adapters. Real-time operational SCADA data is obtained from 1) a legacy SCADA system that is wrapped with a legacy wrapper to provide the necessary component interfaces and 2) an Inter-Control Center Communication Protocol (ICCP) data server connected to other control centers or substations. Both sources of SCADA data publish updates using identical component interfaces as they become available, and applications that need real-time updates subscribe to receive them independent of which source supplies the data. A DMS system external to the control center is also shown and could receive the same SCADA data updates as they arrive by also subscribing. Many other application interactions are supported as well.

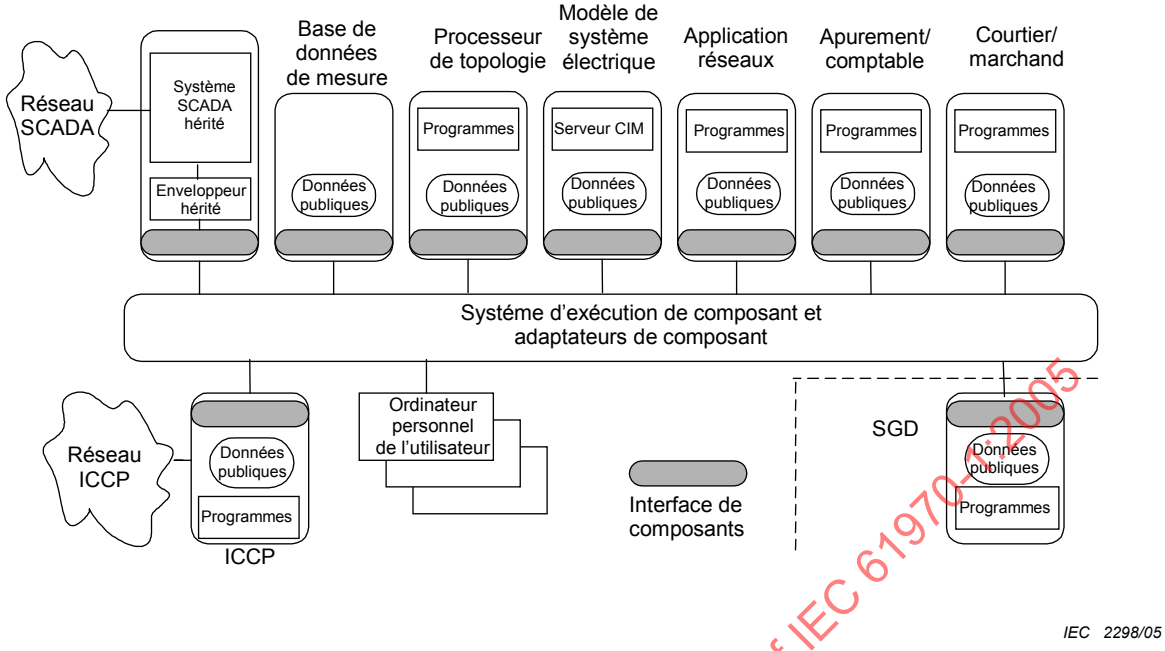

Figure 2 – EMS utilisant des interfaces standard de composants EMS-API

5.13.2 Scénario utilisant des enveloppeurs hérités et des adaptateurs de composants

Un autre exemple est celui d'un fournisseur de produits SCADA qui dispose d'un système existant (par exemple une application d'acquisition des données et de commande SCADA); ce système doit être empaqueté comme un composant CORBA pour être vendu comme une application prête à l'emploi et assurer l'interface avec un système conteneur CORBA. L'application SCADA existante dispose déjà d'une interface de programmation d'application (API) propriétaire, datant de plusieurs années, et dont l'utilisation a été testée et approuvée de manière appropriée dans diverses installations. La réécriture d'une partie de l'application pour en faire un composant CORBA ne constitue pas une justification notable des coûts.

L'approche logique consiste ainsi à présenter cette “application héritée” avec un enveloppeur prenant en charge les interfaces de composants nécessaires. L'enveloppeur présente l'application (ou plus probablement une petite partie de l'application) comme un composant CORBA standard. Cet enveloppeur est alors responsable de la mise en correspondance de l'API propriétaire existante avec celle spécifiée pour le composant. Cette mise en correspondance peut être relativement élaborée.

Considérons maintenant le cas d'un client ayant acheté une application SCADA assemblée à un fournisseur de produits SCADA et souhaitant intégrer cette application dans son entreprise. Actuellement, il est pratiquement certain que la majorité des applications à intégrer à l'application SCADA ne sont pas conformes à un modèle de composants unique et standard, voire à aucun modèle de composants standard, comme dans le cas considéré.

Le client a choisi un intergiciel d'intégration pour faciliter la tâche d'intégration globale. L'une des principales raisons pour lesquelles l'entreprise de service public a acheté le produit intergiciel d'intégration est qu'il facilite l'adaptation des nombreuses interfaces d'application, composants ou autres, aux services communs et au modèle d'échange d'informations fourni par l'intergiciel d'intégration. Le produit intergiciel d'intégration de l'entreprise risque de ne pas prendre en charge directement le modèle de composants utilisé par le composant SCADA ou de ne pouvoir accueillir aucun modèle de composants. Le fournisseur d'intergiciel ou l'intégrateur du système a donc recours à un adaptateur de composant.

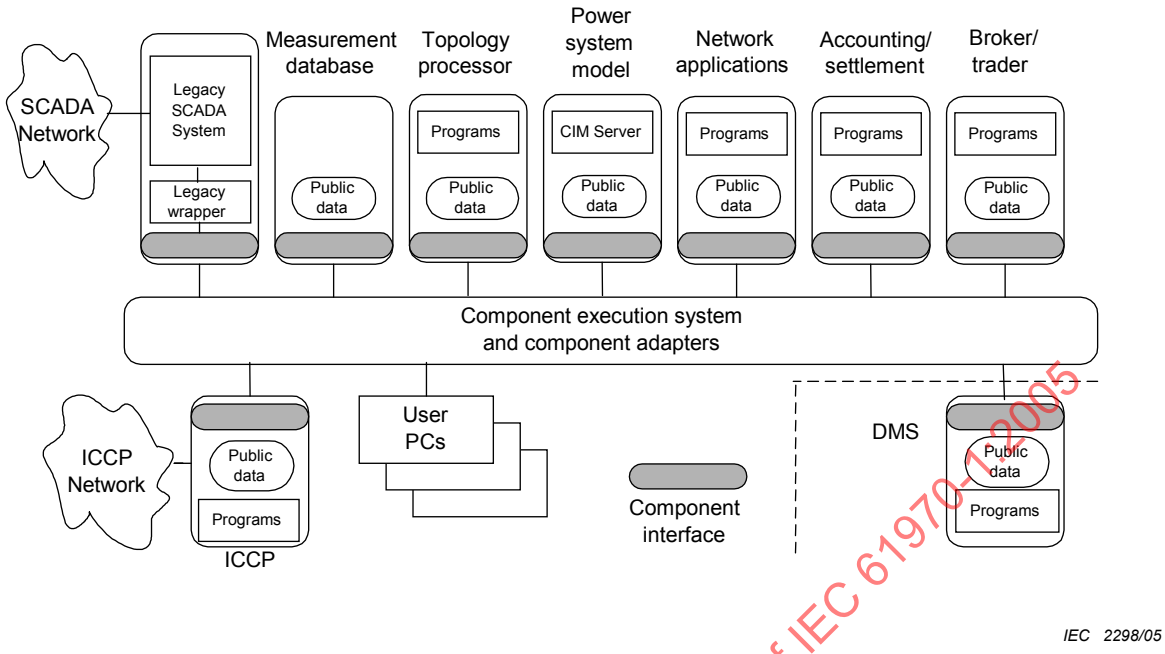

Figure 2 – EMS using EMS-API component standard interfaces

5.13.2 A scenario using both legacy wrappers and component adapters

An alternative example is of a SCADA vendor who has an existing system (for example, a typical SCADA data acquisition and control application) that needs to be packaged as a CORBA component so it can be sold as a plug-and-play application to be interfaced to a CORBA container system. The existing SCADA application already has a proprietary API, which has existed for years, is proven and well tested, and used in a variety of installations. There is little cost justification in rewriting any portion of the application just to make it a CORBA component.

The logical approach then is to front-end this “legacy application” with a wrapper that supports the needed component interfaces. The wrapper presents the application (or more likely a small part of the application) as a standard CORBA component. This wrapper would likely be responsible for mapping the existing proprietary API into that specified for the component. This mapping could be quite elaborate.

Now a customer that has purchased the SCADA vendor's componentized SCADA application wants to integrate it into his enterprise. What is almost certain today is that the majority of the applications to be integrated with the SCADA application will not be compliant to a single standard component model, or for that matter, any standard component model.

The customer has chosen an integration middleware product to help with the overall integration task. One of the primary reasons the utility bought the integration middleware product is that it facilitates the adaptation of the various application interfaces, component or otherwise, to the common services and information exchange model provided by the integration middleware product. The enterprise integration middleware product may not support the component model used by the SCADA component directly, or may not support a component model at all. The middleware vendor or the system integrator thus introduces a component adapter.

Ce scénario met en évidence la différence qui existe entre un enveloppeur hérité et un adaptateur de composant dans le modèle de référence. Dans ce cas, l'enveloppeur relève de la responsabilité du fournisseur SCADA et l'adaptateur de celle du fournisseur d'intergiciels (il est évident que l'entreprise de service public aurait pu choisir l'interface SCADA propriétaire et l'enveloppeur n'aurait alors pas été introduit.) Dans la mesure où l'enveloppeur présente l'application SCADA comme un composant standard, l'adaptateur du fournisseur d'intergiciels peut être normalisé. Les fournisseurs d'intergiciels d'intégration proposent effectivement des adaptateurs standard, généralement pour les systèmes de gestion de base de données relationnels (SGBDR) (par exemple Oracle, langage structuré d'interrogation (SQL) Server, etc.), des applications standard répandues (par exemple SAP, Peoplesoft, etc.) et d'autres ressources courantes (par exemple MQSeries, etc.).

6 Normes EMS-API

6.1 Généralités

Comme indiqué dans le modèle de référence, le modèle d'information commun (CIM) et les Spécifications d'interface de composants (CIS) constituent les parties essentielles du modèle en cours de normalisation. Ces parties s'inscrivent en réalité dans les normes de la série CEI 61970 en cours d'élaboration par le comité d'études 57 de la CEI sous le nom de EMS-API. La structure réelle du document pour la CEI 61970 est la suivante:

Partie 1: Lignes directrices et exigences générales

Partie 2: Glossaire

Partie 3xx: Modèle d'information commun (CIM)

Partie 4xx: Spécifications d'interface de composants (CIS)

Partie 5xx: Mises en correspondance des technologies CIS

6.2 CIM (CEI 61970-3xx)

6.2.1 Généralités

L'exploitation des composants d'application SCADA/EMS/SGD requiert généralement un modèle élaboré comprenant des mesures, une connectivité de réseau, des caractéristiques d'équipement, etc. Le modèle CIM prévoit ce modèle en fournissant des composants d'application avec une appréhension logique globale d'un système électrique. Le CIM est un modèle abstrait représentant tous les principaux objets présents dans une entreprise publique de production d'électricité qui sont généralement contenus dans un modèle d'information EMS et nécessaires dans de multiples applications. Ce modèle comprend des classes publiques et des attributs de ces objets ainsi que les relations existantes entre eux (voir la CEI 61970-301). Ce modèle est décrit dans le langage de modélisation unifié (UML) et conservé comme unique fichier modèle Rational ROSE unifié. D'importantes parties des documents de la CEI 61970-3xx sont autogénérées à partir du fichier modèle utilisant Rational SODA.

Le modèle CIM fait partie intégrante du cadre global de EMS-API. Il fournit un mode standard de représentation des ressources d'un système électrique en classes d'objet et en attributs, ainsi que les relations existantes entre eux, et facilite ainsi l'intégration d'applications EMS développées de manière indépendante par différents fournisseurs, ou entre des systèmes EMS complets développés de manière indépendante, ou encore entre un système EMS et d'autres systèmes ayant trait à différents aspects des opérations du système électrique, telles que la gestion de la production ou de la distribution. Cela est obtenu en définissant un langage commun (c'est-à-dire une sémantique et une syntaxe) fondé sur le modèle CIM pour permettre à ces applications ou systèmes d'accéder aux données publiques et d'échanger des informations indépendamment de la représentation interne des informations.

This scenario illustrates the distinction between a legacy wrapper and a component adapter in the reference model. Here, the wrapper is the responsibility of the SCADA vendor and the adapter, the middleware vendor (obviously the utility could have chosen to use the proprietary SCADA interface, and the wrapper would not even be introduced). To the extent the wrapper presents the SCADA application as a standard component, the middleware vendor's adapter can be standardized. In fact, integration middleware vendors offer standard adapters, typically for Relational Data Base Management Systems (RDBMS) (e.g., Oracle, Structured Query Language (SQL) Server, etc.), popular standard applications (e.g. SAP, Peoplesoft, etc.), and other common resources (e.g., MQSeries, etc.).

6 EMS-API standards

6.1 General

As shown in the reference model, the Common Information Model (CIM) and Component Interface Specification (CIS) are the major parts of the model that are being standardized. In fact, these are parts of the IEC 61970 series of standards being prepared in the IEC Technical Committee 57 under the name EMS-API. The actual document structure for IEC 61970 is as follows:

Part 1: Guidelines and general requirements

Part 2: Glossary

Part 3XX: Common Information Model (CIM)

Part 4XX: Component Interface Specifications (CIS)

Part 5XX: CIS technology mappings

6.2 CIM (IEC 61970-3XX)

6.2.1 General

SCADA/EMS/DMS application components in general require an elaborate model including measurements, network connectivity, device characteristics, etc. to be able to operate. The CIM provides this model, supplying these application components with a comprehensive logical view of a power system. The CIM is an abstract model that represents all the major objects in an electric utility enterprise typically contained in an EMS information model that are needed by multiple applications. This model includes public classes and attributes for these objects, as well as the relationships between them (see IEC 61970-301). This model is described in the Unified Modeling Language (UML) and maintained as single unified Rational ROSE model file. Major portions of the IEC 61970-3XX documents are auto-generated from this model file using Rational SODA.

The CIM is part of the overall EMS-API framework. By providing a standard way of representing power system resources as object classes and attributes, along with their relationships, the CIM facilitates the integration of EMS applications developed independently by different vendors, between entire EMS systems developed independently, or between an EMS system and other systems concerned with different aspects of power system operations, such as generation or distribution management. This is accomplished by defining a common language (i.e., semantics and syntax) based on the CIM to enable these applications or systems to access public data and exchange information independent of how such information is represented internally.

6.2.2 Utilisation prévue du modèle CIM

Les objets représentés dans le modèle CIM sont de nature abstraite et peuvent être utilisés dans des applications très diverses. L'utilisation du modèle CIM va au-delà de son application dans un SCADA/EMS/SGD. Il convient de comprendre la présente norme comme un outil permettant l'intégration dans tout domaine où un modèle de système électrique commun est nécessaire pour faciliter l'interopérabilité et la compatibilité de connexions entre des applications et des systèmes, quelle que soit la mise en oeuvre. Parmi les utilisations spécifiques prévues du modèle CIM figurent:

– Lancer des composants d'application avec des données de configuration.

  Généralement, avant d'être opérationnel, un composant d'application doit être lancé avec les informations courantes d'état et d'événement lors d'une phase de configuration (parfois appelée gestion de données) fondée sur ce modèle. Tout au long de la durée de vie d'un composant d'application, des extensions apportées au système électrique nécessitent de modifier le modèle et donc les données de configuration.

– Réutiliser les données de configuration existantes provenant de systèmes hérités.

– Incorporer des données de configuration existantes provenant de systèmes étrangers.

– Fournir des données de base pour prendre en charge l'échange de données en ligne entre des composants d'application.

  Les données produites par les composants d'application pendant une exploitation en ligne sont présentées à des exploitants et mises à disposition comme des données d'entrée à d'autres applications. Les composants d'application sont également des récepteurs de données provenant d'autres composants d'application. L'échange de données en ligne fait l'objet des normes CIS décrites en 6.3 et le contenu de l'échange de données relève du modèle CIM.

6.2.3 Contexte et temps

La définition du modèle CIM applicable à toutes les entités permet implicitement l'utilisation de multiples instances d'objets à de multiples temps.

Pour établir une distinction entre les entités CIM instanciées dans des composants d'application identiques, il est prévu que les informations échangées entre des composants comportent un contexte ou un environnement. Cela peut être mis en oeuvre comme une chaîne alphanumérique par exemple TRAINING (formation), REALTIME (temps réel), OFFLINE (hors ligne), STUDY (étude) ou REPLAY (réexécutuion). Cependant, il convient que les composants eux-mêmes ignorent le contexte dans lequel ils sont exploités.

Tous les événements d'échange de données transmis entre deux applications basées sur le modèle CIM doivent inclure un concept de temps. Selon l'application concernée, il peut s'agir d'un concept de temps implicite (par exemple le temps est celui de l'application réceptrice) ou explicite (c'est-à-dire que les éléments de données disposent de descripteurs temps, soit individuellement, soit en blocs entiers).

Au sein d'une application, des entités CIM telles que la «PowerSystemResource» ou la «MeasurementValue» (valeur de mesure) peuvent comprendre une date et une heure de départ, une date et une heure de fin et une date et une heure actuelles dans un mode non spécifié. Des applications différentes peuvent gérer le temps de différentes manières, par exemple une variable globale gérée comme un attribut dans des objets individuels, comme un objet associé, etc. Cela est considéré comme relevant de la conception et ne fait donc pas partie du domaine d'application.

6.2.2 Intended use of the CIM

The objects represented in the CIM are abstract in nature and may be used in a wide variety of applications. The use of the CIM goes far beyond its application in an SCADA/EMS/DMS. This standard should be understood as a tool to enable integration in any domain where a common power system model is needed to facilitate interoperability and plug compatibility between applications and systems independent of any particular implementation. Specific anticipated uses of the CIM include:

– Initialize application components with configuration data.

  Typically, before an application component can be made operational, it shall be initialized with current state and event information in a configuration phase (sometimes called data engineering) based on this model. During the lifetime of an application component, extensions are made to the power system requiring changes to the model and thus to the configuration data.

– Reuse existing configuration data from legacy systems.

– Incorporate existing configuration data from foreign systems.

– Provide base data to support on-line exchange of data between application components.

  Data produced by application components during on-line operation is presented to operators and made available as input to other applications. Application components are also receivers of data from other application components. While this on-line data exchange is the subject of the CIS standards described in 6.3, the content of the data exchange is based on the CIM.

6.2.3 Context and time

The CIM definition for all entities implicitly allows multiple instances of objects at multiple times.

To distinguish between the CIM entities instantiated in identical application components, information exchanged between components is expected to include a context or environment. This could be implemented as an alphanumeric string, e.g., TRAINING, REALTIME, OFFLINE, STUDY, and REPLAY. However, the components themselves should not be aware of the context in which they are operating.

All data exchange events between two CIM based applications shall include a concept of time. Depending on the application, this can either be implicit (e.g., the time is the time of the receiving application) or explicit (i.e., data items have time tags, either individually or for entire blocks).

Within an application, CIM entities such as PowerSystemResource or MeasurementValue may include a start date and time, end date and time, and current date and time in an unspecified way. Different applications may hold time in different ways, e.g., a global variable, held as an attribute in individual objects, as an associated object, etc. This is considered to be a design issue and thus not in scope.

En outre, les données d'un système opérationnel seront appelées à évoluer dans le temps. Des objets seront créés ou supprimés et les valeurs de propriété seront modifiées. En conséquence, chaque objet disposera d'une heure de création et, éventuellement, d'une heure de suppression. Les valeurs de propriété auront une ou plusieurs heures correspondant au temps de création ou de modification. Le modèle CIM ne modélise pas explicitement cet aspect, à l'exception de la valeur de mesure qui dispose d'un horodatage explicite pour le temps de la dernière mise à jour. L'absence d'horodatage explicite pour les autres objets et valeurs de propriété n'empêche pas d'utiliser le modèle dans des applications où le temps est enregistré, par exemple dans des applications d'entreposage de données. Pour de telles applications, il est recommandé que:

– tous les objets disposent d'un horodatage de “création” et de "suppression";
– toutes les valeurs de propriété disposent d'un horodatage de "dernière mise à jour";
– tous les horodatages des "dernières mise à jour" des propriétés ainsi que des valeurs correspondantes soient mémorisés dans un historique.

Il est possible d'accéder aux valeurs des données ainsi qu'aux horodatages associés par l'intermédiaire des API tel que spécifié dans les normes CIS de la CEI 61970-4xx.

6.2.4 Cohérence des relations et des attributs

Le modèle CIM représente un ensemble d'entités, de relations et d'attributs d'un réseau électrique dans un état particulier. Le présent document ne spécifie pas la manière dont il convient qu'une application contrôle la cohérence des relations et des attributs des objets instanciés qu'elle comporte. Par exemple, si un état d'inverseur est modifié, il peut être nécessaire de recalculer les relations topologiques et d'obtenir un nouvel ensemble de valeurs de mesure. Cela peut concerner plusieurs applications.

Les interfaces d'application sont chargées de fournir des informations sur l'état de cohérence nécessaires pour pouvoir échanger des informations (par exemple une énumération 'ModelValid', 'TopologyValid', 'LoadFlowValid').

Pour une configuration réseau particulière, l’équipement conducteur peut être regroupé en départ et des noeuds de connectivité peuvent être rassemblés en noeuds de topologie. Le contexte d'échange d'informations a pour rôle de déterminer si les informations se rapportent à "ASBUILT", "CURRENT" etc. Cela vaut autant pour les valeurs de mesure et autres attributs simples que pour les relations topologiques.

Par exemple, une application de base de données peut gérer des instances dans leur état ASBUILT et une application SCADA des instances dans leur état "CURRENT". Le modèle CIM est identique pour les deux applications. Les valeurs des instances au sein de chaque application peut être différent.

6.2.5 Relations avec la série de normes CEI 61968

Comme il a été indiqué précédemment dans l’Article 4.4, la série de normes CEI 61968 s'appuient également sur le modèle CIM en élargissant dans toute la mesure du possible la base de CIM contenu dans la CEI 61970-301 pour y inclure des spécialisations supplémentaires de classes existantes mais également en ajoutant de nouveaux ensembles de classes à des objets modèles existant dans le domaine du problème de la distribution. En conséquence, pour comprendre l'ensemble du domaine d'application du modèle CIM, il est nécessaire de considérer le modèle CIM tel que décrit dans les séries de normes CEI 61970 et CEI 61968 [10].

---

10 Le modèle UML du CIM élargi, qui prend en charge les deux ensembles de normes, est conservé par le groupe de travail 14 du comité d'études 57 de la CEI responsable de la série CEI 61968.

In addition, the data in an operational system will typically evolve over time. Objects will be created or deleted and their property values will be changed. Hence, each object will have a creation time and possibly a deletion time. Property values will have one or more times when they have been created or changed. The CIM does not explicitly model this except for the MeasurementValue that has an explicit time-stamp for the last updated time. The lack of explicit time-stamps for the remaining objects and property values does not prevent the use of the model in applications where time is recorded, e.g. in data warehousing applications. For such applications it is recommended that

– all objects have a “creation” time-stamp and a “deletion” time-stamp;
– all property values have a “last updated” time-stamp;
– a history is maintained for all property “last updated” time-stamps together with the corresponding values.

The data values together with associated time stamps can be accessed through APIs as specified in the IEC 61970-4XX CIS standards.

6.2.4 Consistency of relationships and attributes

The CIM represents a set of entities, relationships and attributes of an electrical network at a particular state. This document does not specify how an application should control consistency of the relationships and attributes of objects instantiated within it. For example, if a switch status is changed, it may be necessary to recalculate the topological relationships and to obtain a new set of measurement values. This may involve more than one application.

Application interfaces are responsible for providing consistency status information required to qualify information exchanges (e.g., an enumeration 'ModelValid', 'TopologyValid', 'LoadFlow Valid').

For a particular network configuration, conducting equipment can be grouped into feeders, and connectivity nodes may be grouped into topology nodes. It is part of an information exchange context to define whether the information refers to 'ASBUILT, 'CURRENT’ etc. This is equally true for measurement values and other simple attributes as it is for the topological relationships.

For example, a database application may hold instances in their ASBUILT state, and a SCADA application would hold instances in their ‘CURRENT’ state. The CIM model is identical for both applications. The instances within each application may have different values.

6.2.5 Relationship to IEC 61968 series of standards

As stated earlier in Clause 4.4, the IEC 61968 series of standards also build on the CIM, by extending the CIM Base contained in IEC 61970-301 wherever possible by extending it to include additional specializations of existing classes, but also adding entirely new sets of classes to model objects found in the distribution problem domain. So to understand the entire scope of the CIM, it is necessary to consider the CIM as described in both the IEC 61970 and IEC 61968 series of standards[10].

[10] The extended CIM UML model supporting both sets of standards is maintained by IEC TC57 WG14, which is responsible for the IEC 61968 series.

6.3 CIS (CEI 61970-4xx)

Les documents CIS de la CEI 61970-4xx ont pour objet de spécifier les interfaces qu'un composant doit utiliser pour faciliter l'intégration avec d'autres composants développés de manière indépendante. Bien que des applications et des fonctions types soient indiquées dans l'Annexe B pour aider à déterminer le type d'informations à transférer, l'objet n'est pas de tenter de définir des composants en tant que tels. Il convient que les fournisseurs de composants soient libres d'empaqueter différents ensembles d'interfaces de composants dans des paquetages de composants tout en restant en conformité avec les normes EMS-API.

Les CIS spécifient les deux principales parties d'une interface de composants:

a) Les interfaces qu'un composant (ou une application) met généralement en oeuvre pour échanger des informations avec d'autres composants (ou applications) et/ou pour accéder de manière standard à des données disponibles au public. Les interfaces de composants décrivent les événements spécifiques, les méthodes et les propriétés que les applications peuvent utiliser pour atteindre ce but.

b) Le contenu des informations ou les messages qu'un composant échange avec d'autres composants. Cela est parfois désigné comme un modèle d'échange d'informations.

La CEI 61970-4xx est structurée de sorte à documenter séparément ces deux parties:

a) Parties 401 à 449:

   Ces parties sont consacrées à la spécification des services génériques que les interfaces de composants doivent prendre en charge. Ces spécifications décrivent la fonctionnalité d'interface qui est normalisée en utilisant des textes explicatifs et des notations en langage de modélisation unifié (UML) et en langage de définition d'interface (IDL). Ces spécifications définissent les services génériques qu'une application peut utiliser pour échanger des informations avec d'autres applications ou pour accéder à des données publiques.

b) Parties 450 à 499:

   Ces parties sont dédiées aux spécifications concernant les exigences spécifiques à l'échange d'informations pour des catégories d'applications types, telles que définies à l'Annexe B. Ces spécifications définissent le contenu des informations des échanges d'informations standard entre les applications. Elles sont définies comme des événements mais peuvent être échangées de différentes manières, y compris être publiées comme des messages, des notifications suivies d'une requête, ou échangées comme des documents XML dans certains cas. Si nécessaire, les propriétés et les méthodes qu'une interface doit prendre en charge sont également identifiées. La documentation de support comprend les cas d'utilisation et les diagrammes de séquences d'événements.

   En fonction du type d'échange prévu pour la catégorie de l'application, des services génériques spécifiques peuvent être spécifiés pour assurer l'interopérabilité entre des composants développés par des fournisseurs différents. Ces normes visent à assurer la meilleure souplesse possible dans le choix de l'intergiciel pour permettre d'échanger effectivement des informations tout en continuant à assurer l'interopérabilité.

   Le contenu des informations est documenté dans un modèle d'échange d'informations (MEI) pour chaque catégorie d'application.

Les spécifications CIS de la CEI 61970-4xx sont documentées indépendamment de la technologie sous-jacente utilisée pour mettre en oeuvre ces spécifications. La série CEI 61970-4xx de documents est ainsi parfois appelée CIS de niveau 1. La série de la CEI 61970-5xx de documents CIS décrite en 6.4 fournit les mises en correspondance des spécifications CIS de la CEI 61970-4xx avec les technologies d'infrastructures spécifiques, et de ce fait est parfois appelée CIS de niveau 2.

6.3 CIS (IEC 61970-4XX)

The purpose of the IEC 61970-4XX CIS documents is to specify the interfaces that a component shall use to facilitate integration with other independently developed components. Although typical applications and functions are identified in Annex B to assist in defining the types of information that shall be transferred, the purpose is not to attempt to define components *per se*. The component vendors should be free to package different collections of component interfaces into component packages while still remaining compliant with the EMS-API standards.

The CIS specifies the two major parts of a component interface:

a) The interfaces that a component (or application) should implement to be able to exchange information with other components (or applications) and/or to access publicly available data in a standard way. The component interfaces describe the specific events, methods, and properties that can be used by applications for this purpose.

b) The information content or messages that a component exchanges with other components. This is sometimes referred to as an information exchange model.

IEC 61970-4XX is organized to separately document these two parts:

a) Parts 401 to 449:

   These are reserved to specify the generic services to be supported by component interfaces. These specifications describe in narrative terms with text, Unified Modelling Language (UML) notation, and IDL the interface functionality that is standardized. These specifications define the generic services that can be used by any application to exchange information with another application or for public data access.

b) Parts 450 to 499:

   These are reserved for specifications that address the specific information exchange requirements for typical application categories, as defined in Annex B. These specifications define the information content of the standard information exchanges between applications. These are defined as events but may be exchanged in a variety of ways, including being published as messages, notifications followed by a request, or exchanged as XML documents in some cases. The properties and methods to be supported by each interface, if required, will also be identified. Supporting documentation includes use cases and event sequence diagrams.

   Depending on the type of exchange envisioned for the application category, specific generic services may be specified to ensure interoperability between components developed by different suppliers. The intent in application of these standards is to provide as much flexibility as possible in the middleware chosen to actually accomplish the information exchanges while still ensuring interoperability.

   The information content is documented in an Information Exchange Model (IEM) for each application category.

The IEC 61970-4XX CIS specifications are documented in a manner that is independent of the underlying technology used to implement them. Thus the IEC 61970-4XX series of documents are sometimes referred to as Level 1 CIS. The IEC 61970-5XX series of CIS documents described in 6.4 provide the mappings for the IEC 61970-4XX CIS specifications to specific infrastructure technologies, and are thus sometimes referred to as Level 2 CIS.

6.4 Mises en correspondance des technologies CIS (CEI 61970-5xx)

En raison de leur conception, les documents CIS sont indépendants de la technologie d'infrastructure sous-jacente; ils doivent donc être mis en correspondance avec des technologies spécifiques pour être mis en oeuvre. Pour assurer l'interopérabilité, une mise en correspondance standard doit être élaborée pour chaque interface à chaque technologie. Par exemple, si Java est la technologie sélectionnée pour la mise en oeuvre, il doit exister une mise en correspondance standard des services de publication et d'abonnement d'événement spécifiés dans le document CIS avec les services Java.

De même, les définitions d'événement compilées dans le MEI à partir des documents CIS pour chaque catégorie d'applications doivent être mises en correspondance avec le langage spécifique utilisé pour transmettre les informations. Par exemple, si un courtier de message est destiné à livrer des messages XML, les événements CIS doivent être mis en correspondance avec le langage XML.

Il est prévu que les mises en correspondance ou spécifications suivantes deviennent des normes associées dans cette série lorsque ces normes seront déployées. Certaines sont spécifiques au langage, d'autres à l'intergiciel:

- langage C++
- langage C
- CORBA®
- DCOM
- Java®
- XML. La spécialisation XML assure l'interopérabilité entre des composants développés de manière indépendante avec une interface GID lorsque la messagerie basée sur le langage XML est utilisée comme technologie d'intégration.

7 Fonctionnalité prévue de l'infrastructure générale

7.1 Généralités

Cet article décrit les capacités d'infrastructure entre les applications dans une entreprise de service public nécessaires pour intégrer des composants distribués. Ces services sont fournis par le système d'exécution de composants décrit dans le modèle de référence EMS-API à l'Article 4. Les services et la fonctionnalité décrits sont indépendants de la technologie sous-jacente utilisée pour cette infrastructure.

Dans le cadre des exigences suivantes, un “événement” est une unité d'échange d'informations émise de manière asynchrone par sa source (“push"). Un “composant” est un module réutilisable d'un logiciel d'application sur un bus d'intégration servant soit d'éditeur soit d'abonné (récepteur) à un échange d'informations.

La liste suivante énumère les capacités qu'il convient de fournir à un système d'exécution de composants pour disposer d'une capacité d'intégration complète entre les applications. Cependant, cela ne signifie pas que toutes les mises en oeuvre doivent prendre en charge chaque fonction. La liste est fournie comme point de départ pour la préparation d'une spécification pour un déploiement de système réel de ces normes. L'importance des exigences spécifiques pour un déploiement dépend dans une large mesure des besoins spécifiques opérationnels et de performance des systèmes interconnectés.

Il convient qu'un système d'exécution de composants:

a) permette aux composants d'échanger des informations dont la complexité est arbitraire;

6.4 CIS technology mappings (IEC 61970-5XX)

Since the CIS documents are independent by design of the underlying infrastructure technology, they shall be mapped to specific technologies for implementation purposes. To ensure interoperability, there shall be a standard mapping for each interface to each technology. For example, if Java is the chosen implementation technology, then there needs to be a standard mapping of the publishing and event subscription services specified in the CIS document to Java services.

Similarly, the event definitions compiled in the IEM from the CIS documents for each application category need to be mapped to the specific language used for transmission of the information. For example, if a message broker is to be used to deliver XML messages, then the CIS events need to be mapped to XML.

It is anticipated that the following mappings or specializations will become companion standards in this series as these standards are deployed. Some are language-specific and some are middleware-specific:

- C++ language;
- C language;
- CORBA®;
- DCOM;
- Java®;
- XML. The XML specialization will provide interoperability between independently developed components with a GID interface when XML based messaging is used as the integration technology.

7 General expected infrastructure functionality

7.1 General

This clause describes utility inter-application infrastructure capabilities necessary to integrate distributed components. These services are provided by the Component Execution System described in the EMS-API reference model in Clause 4. The services and functionality described are independent of the underlying technology used for this infrastructure.

In the context of the following requirements, an “event” is a unit of information exchange which is issued asynchronously by its source (“push”). A “component” is a reusable module of application software on an integration bus serving as either a publisher or subscriber (receiver) of an information exchange.

The following is a list of capabilities that should be provided by the Component Execution System to have a full-featured inter-application integration capability. However, this is not meant to imply that every implementation shall support each and every function. The list is provided as a starting point in preparing a specification for an actual system deployment of these standards.

The specific requirements of importance to a deployment will depend to a large extent on the specific operational and performance needs of the interconnected systems.

A Component Execution System:

a) should allow components to exchange information of arbitrary complexity;

b) puisse être mis en oeuvre en utilisant diverses formes de technologies de composants distribués (par exemple CORBA, DCOM, courtiers de message, intergiciel orienté message, bases de données relationnelles, bases de données orientées objet et autres) (voir Article 6);

c) permette le déploiement des composants de publication et/ou d'abonnement par des administrateurs de système indépendamment des autres composants;

d) garantisse la possibilité de réutiliser entièrement des informations publiées, c'est-à-dire une fois qu'un type d'événement donné est édité, toute nouvelle entité autorisée peut acquérir l'événement sans devoir effectuer de modification ni d'ajout au composant de publication;

e) fournisse une fonction d'historique des événements génériques comme un composant. Cela permet de sauvegarder la totalité ou une sélection des échanges d'informations dans une mémoire permanente;

f) prenne en charge un schéma d'historique des événements basé sur un modèle de métadonnées pour l'échange d'informations;

g) fournisse un composant d'historique des événements qui enregistre l'heure d'émission de chaque événement par le composant de publication;

h) soit capable de prendre en charge des versions de modèles d'informations événements et des versions de composants. (Cela permet de conserver une vérification rétrospective exhaustive capable de prendre en charge une reconstruction rigoureuse de l'historique, si besoin est);

i) fournisse un composant superviseur entre les applications qui analyse l'état de toutes les interfaces de composants d'application connectées à des services d'une entreprise de service public. Il peut être activé et désactivé et fournir des capacités de surveillance des performances. Ces éléments aident à fournir des statistiques servant à identifier les goulots d'étranglement ou les zones sujettes à des améliorations futures. Les informations sont nécessaires pour aider les administrateurs à configurer les informations échangées parmi les composants et pour en assurer la disponibilité;

j) permette à un composant d'envoyer ou de demander des informations sans connaître l'emplacement physique du composant récepteur ni savoir s'il est réellement connecté. Le récepteur peut ne pas être joignable en raison d'un problème de réseau ou peut être déconnecté naturellement comme dans le cas d'utilisateurs mobiles qui ne se connectent que périodiquement. Les composants peuvent être indisponibles en raison d'un échec ou du fait qu'ils ne fonctionnement que pendant certaines heures. Les informations en attente doivent être transmises lorsque le réseau redevient disponible ou lorsque l'application réceptrice est prête à traiter les requêtes.

Les paragraphes suivants fournissent les capacités générales prévues pour les différentes parties du système d'exécution de composants EMS-API.

7.2 Conteneur de composant

Les conteneurs de composants fournissent un ensemble d'API pour composants pour appeler les services pris en charge par les conteneurs. Les services suivants sont généralement fournis comme faisant partie des conteneurs de composants de série disponibles dans le commerce. Les services informatiques distribués couramment et nécessaires pour prendre en charge l'EMS-API sont énumérés ci-après:

a) Service de cycle de vie du composant: il permet de démarrer, arrêter et commander les composants à exécuter.

b) Service de dénomination: il fournit un service de dénomination de composants qui prend en charge une structure hiérarchique et permet à un composant de repérer d'autres composants en utilisant un nom interprétable par l'utilisateur. Le service de dénomination prend en charge l'utilisation de noms existants de l'entreprise de service public, ainsi que la création, la suppression et l'attribution de noms.

c) Service d'horloge: il permet à tous les composants distribués d'avoir la même heure avec une exactitude configurable.

b) should be able to be implemented using various forms of distributed component technology (e.g., CORBA, DCOM, message brokers, message oriented middleware, relational databases, object-oriented databases, or others) (refer to Clause 6);

c) should allow publisher and/or subscriber components to be deployed by system administrators independently of other components;

d) should ensure that published information is completely re-usable in the sense that once a given type of event is published, any new authorized entity may acquire the event without having to make any changes or additions in the publisher component;

e) should provide a generic event history facility as a component. This allows all or selected information exchanges to be saved in a permanent store;

f) should support an Event History schema based on a metadata model for information exchange;

g) should provide an Event History component to record the time at which the publishing component issued each event;

h) should be capable of supporting event information model versions and component versions. (This allows a complete audit trail to be preserved which is capable of supporting rigorous reconstruction of history, if that should become a requirement.);

i) should provide an Inter-application Supervisor component that analyzes the state of any application component interface connected to the utility services. It may be enabled and disabled, and has the capability to provide performance monitoring capabilities. Those elements will help to provide statistics in order to identify bottlenecks or areas subject to improvement in the future. The information is required to help the administrators configure information exchanged among components and to ensure availability;

j) should allow a component to send or request information without knowing where the receiving component is physically located or if it is currently connected. The receiver may be unreachable because of a network problem, or be naturally disconnected as in the case of mobile users who only connect periodically. Components may be unavailable because they have failed or because they only run during certain hours; when the network becomes available or the receiving application is ready to process requests, the waiting information shall be delivered.

The following subclauses provide general capabilites expected for the different parts of the EMS-API component execution system.

7.2 Component Container

Component Containers provide a set of APIs for components to invoke the services supported by the containers. The following services are generally provided as part of commercially-available off-the-shelf Component Containers. The list of such common distributed computing services needed to support the EMS-API are:

a) Component Life Cycle Service: this service allows the starting, stopping, and control of components to be executed.

b) Naming Service: this service provides a component naming service that supports a hierarchical structure and allows a component to locate other components using a human readable name. The naming service supports the use of existing utility names, as well as the creation, removal and aliasing of names.

c) Time Service: this service provides a way for distributed components to all have the same time with a configurable accuracy.

d) Service de contrôle d'accès simultané: il facilite la gestion d'éléments partagés similaires qui sont distribués, par exemple lorsque plusieurs composants s'occupent de différentes parties (échanges, types d'échange) d'un même objet métier dans la vie réelle.

e) Service de sécurité: il permet à une application de déterminer et de vérifier le niveau d'importance des composants et des utilisateurs avec lesquels des échanges sont effectués, ainsi que le cryptage et le décryptage d'échanges particuliers. Le service propose également une authentification de l'hôte, c'est-à-dire l'authentification du ou des noeuds.

f) Service de transaction: il permet à une application de déclarer le début et la fin d'une transaction multi-étape qui réussit ou échoue en tant qu'unité atomique.

g) Service d'interaction des composants: il assure la fiabilité du transfert de messages avec une qualité de service réglable. Il assure également la gestion du cycle de vie des services d'interaction (création, suppression, copie et déplacement) et l'interrogation de l'interaction établie (principalement valable pour des interactions de type publier/s'abonner).

h) Service de messagerie publier/s'abonner: il assure un transfert de messages synchrone et asynchrone entre des instances de composants découplées (anonymes).

i) Service de messagerie demander/répondre: il assure la fiabilité du transfert de messages entre des instances de composants couplées identifiées.

j) Service de messagerie publier/répondre: il fournit une initiation découplée d'un transfert de messages (Publier) qui s'achève par un transfert couplé (Répondre).

k) Service de flux des travaux: il permet aux programmeurs d'application de créer et d'exécuter des agents d'automatisation des processus métiers.

7.3 Intergiciel

Les services intergiciels sont nécessaires pour prendre en charge les conteneurs de composants avec des services fondamentaux requis pour les applications distribuées quelle que soit l'utilisation des modèles de composants. La délimitation entre l'intergiciel et les conteneurs de composants est plus abstraite que concrète.

7.4 Services de profil de communication

L'intégration de deux composants nécessite la présence d'une connexion entre eux. Dans la mesure où il existe plusieurs types de réseau, des ressources différentes emploient des protocoles différents, tels que IIOP et le protocole de transfert hypertexte (HTTP). Pour connecter plusieurs composants, un système d'intégration doit faire concorder de manière transparente et pour les composants les différences de réseau et d'intégration.

Conformément aux termes de la CEI 61970, il convient que les services de profil de communication:

a) garantissent la livraison des messages réseau à leur destination réseau (si elle est active);

b) fournissent une livraison garantie en assurant que les messages réseau sont livrés une fois précisément, indépendamment des défaillances ou modifications du réseau;

c) fournissent un ordonnancement garanti, en conservant la séquence d'envoi de la source lors de la livraison de messages, indépendamment des défaillances ou modifications du réseau;

d) garantissent qu'un message de non livraison soit envoyé à la source si un message réseau ne peut pas être livré à la destination;

e) fournissent une qualité de service réglable pour sélectionner la priorité des messages réseau ou un chemin réseau spécifique pour une livraison;

f) fournissent une adaptation dynamique à la vitesse de traitement des messages réseau par la destination réseau pour permettre le fonctionnement des destinations lentes sur les services.

d) Concurrency Control Service: this service facilitates the management of shared, similar items that are distributed, for example when multiple components take care of different parts (exchanges, exchange types) of the same business object in real life.

e) Security Service: this service allows an application to set and verify the privilege level of components and users with which exchanges are being performed, as well as encryption and decryption of individual exchanges. This service also supplies host authentication, i.e., authentication of node(s).

f) Transactional Service: this service allows an application to declare the beginning and end of a multi-step transaction that either succeeds or fails as an atomic unit.

g) Component Interaction Service: this service provides reliable message transfer with a selectable Quality of Service. It also provides life-cycle management of interaction services (create, delete, copy, and move) and querying of established interaction (mainly valid for Publish and Subscribe interactions).

h) Publish and Subscribe Messaging Service: this service provides synchronous and asynchronous message transfer between decoupled (anonymous) component instances.

i) Request/Reply Messaging Service: this service provides reliable synchronous message transfer between coupled, identified component instances.

j) Publish/Reply Messaging Service: this service provides decoupled initiation of a message transfer (Publish), which is then finished by a coupled transfer (Reply).

k) Workflow Service: this service allows application level programmers to create and run business process automation agents.

7.3 Middleware

Middleware services are needed to support the component containers with fundamental services required for distributed applications independent of the use of component models. The demarcation between middleware and component containers is abstract rather than concrete.

7.4 Communication Profile Services

Integrating two components requires a connection between them. Since there is more than one kind of network, different resources speak different protocols, such as IIOP and Hyper-Text Transport Protocol (HTTP). To connect multiple components, an integration system shall reconcile network and protocol differences transparently to the components.

IEC 61970 series of standards requires that the Communication Profile services should:

a) guarantee delivery of network messages to their network destination (if active);

b) provide guaranteed delivery, ensuring that network messages are delivered exactly once, regardless of network failures or changes;

c) provide guaranteed ordering, preserving the sending sequence of the source when delivering messages, regardless of network failures or changes;

d) guarantee that if a network message cannot be delivered to a network destination, the network source will receive a message indicating the non-delivery;

e) provide a selectable quality of service for prioritization of network messages or delivery via specific network paths;

f) provide dynamic adaptation to the speed of processing network messages by the network destination to allow slow destinations to work on the services.

7.5 Services spécifiques à une entreprise de service public

Il existe plusieurs services spécifiques à une entreprise de service public pouvant être nécessaires pour prendre en charge des composants dans un EMS et qui ne sont pas proposés par des systèmes d'exécution de composants disponibles dans le commerce. L'Annexe C fournit une liste d'exemples de services potentiels appartenant à cette catégorie et des solutions alternatives pour fournir ces services. Il est prévu que si l'un de ces services est requis pour prendre en charge l'échange d'informations entre des applications, il sera identifié dans le processus d'élaboration de la série CIS de la CEI 61970-4xx et fera partie intégrante de ces normes.

7.5 Utility-specific services

There are a number of utility-specific services that may be needed to support components in an EMS that are not available from commercially-available component execution systems. A list of possible services that fall in this category are listed in Annex C as examples along with alternatives for providing such services. It is expected that if any of these services are required to support information exchange between applications, they will be identified in the process of developing the IEC 61970-4xx series CIS and will become part of those standards.

Annexe A
(informative)

Modèles de composants

A.1 Modèle de composants

Un modèle de composants définit l'architecture de base d'un composant en spécifiant la structure de ses interfaces et les mécanismes d'interaction avec le composant, son conteneur et d'autres composants. Le modèle de composants fournit des lignes directrices pour créer et mettre en oeuvre des composants qui peuvent fonctionner ensemble pour former une plus grande application.

Il existe quatre principaux modèles de composants largement répandus au sein de l'industrie du logiciel actuelle; ils sont décrits dans les articles suivants.

A.2 Enterprise JavaBeans®[11]

La spécification Enterprise JavaBeans (EJB) définit un modèle de composants serveur pour JavaBeans® [2][12]. Les EJB sont des JavaBeans spécialisés et non visuels qui tournent davantage sur serveur que sur client. Un EJB peut être assemblé à d'autres composants logiciels (bean) pour créer une nouvelle application. Un composant logiciel d'entreprise ("enterprise bean") peut être manipulé et personnalisé par l'intermédiaire de sa table de propriétés et méthodes de personnalisation.

Le modèle de composants EJB définit les relations entre un composant logiciel "enterprise bean" et un système de conteneur de composants logiciels "enterprise bean". Les EJB ne requièrent pas l'utilisation d'un système conteneur spécifique. Tout système d'exécution d'application peut être adapté pour prendre en charge des EJB en ajoutant un support pour les services définis dans la spécification. Les services établissent un contrat entre un composant logiciel "enterprise bean" et le conteneur, créant ainsi une couche de transférabilité. Cela correspond à l'adaptateur de composant pour les EJB nécessaire à tout système d'exécution, à l'exception du système d'exécution Enterprise JavaBeans, désigné comme "Enterprise JavaBeans Server" (serveur EJB). Cela permet à un "enterprise bean" de fonctionner dans tous les systèmes d'exécution ou conteneurs prenant en charge les contrats EJB.

Le serveur EJB fournit un ensemble standard de services pour prendre en charge des composants "enterprise bean". Les EJB étant transactionnels, le serveur EJB doit fournir l'accès à un service de gestion des transactions distribuées. Il doit également fournir un conteneur pour l'"enterprise bean", désigné comme conteneur Enterprise JavaBeans (conteneur EJB). Le conteneur EJB met en oeuvre les services de gestion et de commande pour une des classes EJB. Il fournit également les services suivants:

– gestion du cycle de vie;
– commande de transaction;
– gestion de persistance;
– services de distribution transparents;
– services de sécurité.

---

11 Enterprise JavaBeans est l'appellation commerciale d'un produit distribué par Sun Microsystems. Cette information est donnée à l'intention des utilisateurs de la présente Norme internationale et ne signifie nullement que la CEI approuve ou recommande l'emploi exclusif du produit ainsi désigné. Des produits équivalents peuvent être utilisés s'il est démontré qu'ils conduisent aux mêmes résultats.

12 Les chiffres entre crochets se réfèrent à la bibliographie.

Annex A
(informative)

Component models

A.1 Component model

A component model defines the basic architecture of a component, specifying the structure of its interfaces and the mechanisms by which it interacts with its container and with other components. The component model provides guidelines to create and implement components that can work together to form a larger application.

There are four primary component models with wide acceptance within the software industry today that are described in the following clauses.

A.2 Enterprise JavaBeans®[11]

The Enterprise JavaBeans (EJB) specification defines a server component model for JavaBeans® [2][12]. Enterprise JavaBeans are specialized, non-visual JavaBeans that run on a server rather than a client. An Enterprise JavaBean can be assembled with other beans to create a new application. An enterprise bean can be manipulated and customized through its property table and customization methods.

The Enterprise JavaBeans component model defines the relationship between an enterprise bean component and an enterprise bean container system. Enterprise JavaBeans do not require the use of any specific container system. Any application execution system can be adapted to support Enterprise JavaBeans by adding support for the services defined in the specification. The services define a contract between an enterprise bean and the container, thus creating a portability layer. This corresponds to the component adapter for Enterprise JavaBeans that is required for any execution system except the Enterprise JavaBeans execution system, called an Enterprise JavaBeans Server (EJB server). This permits any enterprise bean to run in any application execution system or container that supports the Enterprise JavaBeans contracts.

The EJB server provides a standard set of services to support enterprise bean components. Since Enterprise JavaBeans are transactional, the EJB server should provide access to a distributed transaction management service. It should also provide a container for the enterprise bean, called an Enterprise JavaBeans container (EJB container). The EJB container implements the management and control services for a class of Enterprise JavaBeans classes. It also provides the following services:

– life-cycle management;
– transaction control;
– persistence management;
– transparent distribution services;
– security services.

---

11 Enterprise JavaBeans is the trade name of a product supplied by Sun Microsystems. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

12 Numbers in square brackets refer to the bibliography.

A.3 Composants CORBA

CORBA a défini un modèle de composants comparable à la spécification EJB [3]. Le modèle de composants CORBA® définit:

a) un composant agissant comme un fournisseur d'une ou de plusieurs interfaces CORBA. Un composant:
   - peut fournir et utiliser une ou plusieurs interfaces CORBA individuelles;
   - peut émettre ou consommer un ou plusieurs événements spécifiques;
   - a des propriétés configurables de conception et de temps;
   - a des attributs d'exécution;
   - annonce une interface usine pouvant être utilisée pour créer des instances du composant.

b) un modèle de conteneur pour introduire des services système dans l'environnement d'exécution d'un composant CORBA. Le modèle de conteneur spécifie:
   - des interfaces localisées pour l'interaction d'un composant avec son conteneur;
   - une version simplifiée des transactions CORBA;
   - un support cycle de vie pour optimiser l'utilisation des ressources au sein d'un processus;
   - des politiques de sécurité qui fournissent une autorisation fondée sur l'identité et la délégation client tel que décrit par le service de sécurité CORBA;
   - une gestion de l'état persistant.

Un schéma de mise en correspondance avec EJB permet à un composant EJB d'être pris en charge comme un composant CORBA au sein d'un conteneur fournissant l'activation, les transactions, la sécurité et la persistance. Pour prendre en charge des EJB naturellement au sein d'un conteneur CORBA, il est nécessaire de procéder à une transformation entre les API Java d'EJB et les schémas de mise en correspondance Java des API CORBA correspondantes. Réciproquement, il convient qu'un composant CORBA qui se limite à la fonctionnalité définie par la spécification EJB et est écrit en langage Java puisse être déployé dans un conteneur EJB.

Les conteneurs CORBA sont des contextes d'exécution qui fournissent des interfaces avec les clients et des interfaces bidirectionnelles (liées à des contraintes de localité) avec les instances de composants analysant les services système pour les mises oeuvre de composants. Des conteneurs fournissent des politiques et/ou des API pour prendre en charge des transactions, la sécurité, la gestion d'état et la persistance. Par exemple, des politiques de transaction et de sécurité sont définies dans le descripteur de déploiement du composant. Le conteneur utilise ces descriptions pour effectuer les appels appropriés au service de transaction CORBA et de sécurité CORBA avant d'appeler l'opération requise sur le composant.

Un conteneur fournit les fonctions suivantes pour son composant:

- Toutes les instances de composants sont créées et gérées à l'exécution par son conteneur.
- Certaines métadonnées (par exemple transactions, sécurité, événements et persistance) sont séparées de la mise en oeuvre du composant pour permettre leur manipulation avant le déploiement.
- Un composant peut être personnalisé en éditant ses métadonnées.
- Les accès client aux interfaces fournies par un composant sont délégués par le conteneur dans lequel le composant est déployé.
- Les conteneurs fournissent un ensemble standard de services système à un composant pour lui permettre d'être accueilli par différentes mises en oeuvre de composants.

A.3 CORBA components

CORBA has defined a component model comparable to the EJB specification [3]. The CORBA® component model defines:

a) A component, which acts as a provider of one or multiple CORBA interfaces. A component:
   - can provide and use one or more individual CORBA interfaces;
   - can emit or consume one or more specific events;
   - has configurable design-time properties;
   - has runtime attributes;
   - declares a factory interface that can be employed to create instances of the component.

b) A container model for introducing system services into the runtime environment of a CORBA component. The container model specifies:
   - locality constrained interfaces for a component to interact with its container;
   - a simplified version of CORBA transactions;
   - lifecycle support to optimize resource usage within a process;
   - security policies which provide authorization based on client identity and delegation as described by the CORBA security service;
   - persistent state management.

A mapping to EJB makes it possible for EJB to be supported as a CORBA component within a container that provides activation, transactions, security and persistence. To support EJBs natively within a CORBA container, a transformation between the EJB Java APIs and the Java mappings of the corresponding CORBA APIs is required. Conversely, a CORBA component that restricts itself to the functionality defined by the EJB specification and is written in Java should be deployable in an EJB container.

CORBA Containers are run-time contexts that provide interfaces to clients and bi-directional (locality constrained) interfaces to component instances that abstract system services for component implementations. Containers will provide policies and/or APIs to support transactions, security, state management and persistence. For example, transaction and security policies are defined in the component's deployment descriptor. The container uses these descriptions to make the proper calls to the CORBA transaction service and to CORBA security before invoking the requested operation on the component.

A container provides the following functions for its component:

- All component instances are created and managed at runtime by its container.
- Certain metadata (e.g., transactions, security, events, and persistence) are separated from the component's implementation to allow them to be manipulated prior to deployment.
- A component can be customized by editing its metadata.
- Client access to interfaces provided by a component are delegated by the container in which the component is deployed.
- Containers provide a standard set of system services to a component, enabling the component to be hosted by different container implementations.

A.4 Microsoft® COM/DCOM

Le modèle d'objet de composants (COM) de Microsoft a été initialement élaboré pour prendre en charge l'intégration d'applications de bureau sur la plate-forme Windows [4]. COM est le résultat des efforts déployés par Microsoft pour améliorer OLE (Object Linking and Embedding – liaison et incorporation d'objet) de la version 1 à la version 2. A l'origine, COM ne prenait pas en charge les environnements distribués. La version de COM prenant en charge la distribution est appelée DCOM.

Le problème résolu par COM/DCOM doit faciliter le découpage d'un système logiciel constitué d'une ou de plusieurs classes en paquetages distincts pouvant être développés et maintenus indépendamment les uns des autres.

Le support de composants compris dans COM est relativement simple et comprend deux grandes parties:

1) un langage de description d'interface appelé "Microsoft Interface Definition Language" (MIDL) et des outils de prise en charge de la traduction du code MIDL en description d'interface C++ ou Visual Basic;
2) un environnement d'exécution prenant en charge la consultation et l'activation de composants.

DCOM ajoute ensuite un support de communication à distance à l'environnement d'exécution.

Pour améliorer COM/DCOM, Microsoft a ajouté un ensemble d'interfaces standard montrant des fonctions système diverses et des services tels que les rappels automatiques (objets interfaçables), la persistance, un service de dénomination simple (monikers), librairie type et appel d'interface dynamique (OLE Automation), etc. Ces interfaces, le support d'exécution correspondant et COM/DCOM constituent OLE 2.0.

COM est désormais étendu à un système d'exécution de composants complet appelé COM+. COM+ prend en charge le développement de composants simplifié, l'installation de composant simplifiée, des opérations de «transaction» des événements, la publication et l'abonnement et la mise en file d'attente de messages.

Avec COM+, Microsoft définit un paquetage logique et physique. Le paquetage logique consiste à fractionner les classes et les interfaces en espaces de noms distincts, en réduisant ainsi les collisions de noms de classes interprétables par l'utilisateur. Le paquetage physique est l'unité de déploiement correspondant au composant ou module réel, généralement un ou plusieurs programmes d'exécution (.EXE) ou des modules de chargement de liens dynamiques (.DLL). Dans le cas où un DLL met en oeuvre des classes (à l'inverse des DLL mandataires purs), le chargement peut être considéré comme un conteneur, notamment si le chargement ne contient qu'une fonctionnalité de base ou générale. En règle générale, un composant ou un module met en oeuvre une ou plusieurs classes et interfaces. La mise en oeuvre de classes ou d'interfaces par des composants ou des modules n'est soumise à aucune restriction. Elle relève du choix du réalisateur.

COM/DCOM est également disponible sur d'autres plates-formes informatiques telles que Digital Unix et Solaris. Le support de base pour COM/DCOM est relativement transférable. La fonctionnalité associée au bureau Windows (par exemple, la base de registre) est moins transférable.

A.5 Services Web XML et Microsoft .NET

Les services Web XML fournissent un modèle d'intégration basé sur Internet pour assurer une intégration sans contrainte. Les services Web XML permettent aux applications de communiquer et de partager des données sur Internet, indépendamment du système d'exploitation ou du langage de programmation. Ils sont comme des composants.

A.4 Microsoft® COM/DCOM

Microsoft's Component Object Model (COM) was originally developed to support the integration of desktop applications on the Windows platform [4]. COM came out as one of the results of Microsoft's effort to improve OLE (Object Linking and Embedding) from version 1 to version 2. COM originally did not support distributed environments. The version of COM supporting distribution is called DCOM.

The problem solved by COM/DCOM is to facilitate partitioning a software system consisting of one or more classes into separate packages that can be developed and maintained independently of each other.

The component support in COM is fairly simple and consists of two major parts:

1) an interface description language called Microsoft Interface Definition Language (MIDL) and tools supporting the translation of MIDL code into C++ or Visual Basic interface descriptions;
2) a runtime environment supporting look up and activation of components.

DCOM then adds remote communication support to the runtime environment.

On top of COM/DCOM, Microsoft has added a set of standard interfaces exposing various system functions and services such as callbacks (connectable objects), persistency, a simple naming service (monikers), type library and dynamic interface invocation (OLE Automation), etc. These interfaces, their corresponding runtime support and COM/DCOM are what makes OLE 2.0.

COM is now being extended to a complete component execution system called COM+. COM+ supports simplified component development, simplified component installation, transactioning, events, publish and subscribe, and message queuing.

With COM+, Microsoft defines logical and physical packaging. Logical packaging means splitting up the classes and interfaces into separate namespaces, thus reducing human-readable class name collisions. Physical packaging is the unit of deployment corresponding to the actual component or module, typically one or more executables (.EXE) or dynamically linked load modules (.DLL). In the case where a DLL implements classes (in contrast to pure proxy DLLs), the process loading can be regarded as a container, especially if the loading process only contains basic or framework functionality. A component or module typically implements one or more classes as well as interfaces. There are no restrictions as to how classes or interfaces are implemented by components or modules. This is the choice of the implementer.

COM/DCOM is also available on other computer platforms like Digital Unix and Solaris. The low level support for COM/DCOM is fairly portable. Functionality associated with the Windows desktop (e.g., the registry) is less portable.

A.5 XML Web Services and Microsoft .NET

XML Web Services provide an Internet-based integration model for any-to-any integration. XML Web services allow applications to communicate and share data over the Internet, regardless of operating system or programming language. They are like components.

Les services Web XML fournissent un modèle d'intégration basé sur Internet pour assurer une intégration sans contrainte. Les principaux services sont:

- UDDI (Universal Description and Discovery Information) – Publier et découvrir d'autres services.
- XML et HTTP – Communications et contenu de données.
- SOAP (Simple Object Access Protocol) – Service basé sur des transactions pour échanger des informations et appeler des services par l'intermédiaire d'applications distribuées.

.NET est la stratégie et l'offre de Microsoft pour englober les services Web XML en tous lieux; il s'agit essentiellement d'une plate-forme pour les services Web XML. Les cinq domaines clés suivants doivent être traités pour créer une plate-forme services Web XML:

a) Clients – tous les types d'ordinateurs et d'équipements intelligents capables de comprendre le langage XML.

b) Services – tels que le service d'authentification de Microsoft .NET Passport offrant la capacité d'ouvrir une session individuelle sur tous les sites Web.

c) Serveurs – suite intégrée de programmes pour l'exécution, la gestion et l'orchestration des services Web et logiciels d'application pouvant mémoriser, acheminer, transformer et faire la passerelle vers des données héritées.

d) Outils développeurs – tels que Visual Studio .NET et .NET Framework.

e) Expériences et solutions – associe les meilleures applications locales aux services Internet.

.NET est la plate-forme logicielle de Microsoft capable de traiter la totalité de ces domaines.

La plate-forme Microsoft .NET® comprend une gamme exhaustive de produits basés sur le langage XML et des normes industrielles Internet qui traitent de chaque aspect concernant le développement, la gestion, l'utilisation et l'expérience des services Web XML.

.NET Framework est un environnement d'exécution d'application haute productivité, fondé sur des normes et multi-langage qui prend en charge l'administration et facilite le déploiement. Il fournit un environnement d'exécution d'applications qui gère la mémoire, traite les problèmes liés au contrôle des versions et améliore la fiabilité, l'extensibilité et la sécurité d'une application. .NET Framework est constitué de plusieurs parties, y compris le Common Language Runtime, un ensemble riche de bibliothèques de classes pour construire des services Web XML et ASP.NET.

XML Web Services provide an Internet-based integration model for any-to-any integration. The main services are:

- UDDI (Universal Description and Discovery Information) – Publish and find other services.
- XML and HTTP – Communications and data content.
- SOAP (Simple Object Access Protocol) – Transaction-based service for exchanging information and invoking services across distributed applications.

.NET is Microsoft's strategy and offerings to embrace XML Web Services everywhere, and is essentially a platform for XML Web Services. There are five key areas that need to be addressed to create an XML Web Services platform:

a) Clients – all types of PCs and smart devices that are XML-aware.

b) Services – such as the Microsoft .NET Passport Authentication service that offers single sign-on capability for any Web site.

c) Servers – Integrated suite for running, managing, orchestrating Web services and applications that can Store, route, transform, and bridge to legacy data.

d) Developer tools – such as Visual Studio .NET and the .NET Framework.

e) Experiences and solutions – combine the best of local applications and Internet services.

.NET is Microsoft’s software platform that addresses all of these areas.

The Microsoft .NET® Platform includes a comprehensive family of products, built on XML and Internet industry standards, that provide for each aspect of developing, managing, using, and experiencing XML Web Services.

The .NET Framework is a high-productivity, standards-based, multi-language application execution environment that provides essential infrastructure services and eases deployment. It provides an application execution environment that manages memory, addresses versioning issues, and improves the reliability, scalability, and security of your application. The .NET Framework consists of several parts, including the Common Language Runtime, a rich set of class libraries for building XML Web Services, and ASP .NET.

Annexe B
(informative)

Applications et fonctions types

Tableau B.1 – Applications et fonctions types

Catégorie d'application			Acteur (Exploitant)
	Nom de l'application	Commentaire/explication	
SCADA			Exploitant de terrain
	Synchronisation temporelle	Synchronise les descripteurs temps RTU	
	Communication RTU et PLC	Acquisition de données en exploitation, en utilisant RP570, CEI 60870-5-101, …	Exploitant de poste
	Communication du centre à distance	Communications entre des sites en utilisant ICCP, ELCOM, …	
	Traitement des signaux	Filtrage, échelle, conversion des unités d'ingénierie, vérification de la qualité, vérification des limites	
	Traitement des données	Calculs généralisés, état, analogique et valeur d'accumulateur, traitement qualité	
	Etiquetage	L'étiquetage est utilisé pour attribuer des étiquettes à des objets du monde réel arbitraires représentés dans un SCADA	
	Schémas unilignes	Affichage de processus graphique basé sur une page	
	Listes d'états	Rapport d'état des objets SCADA/EMS	
	Conduite	Commande d'équipement et de séquence (enclenché ou non enclenché)	
	Commande de système		
	Commande inter-sites		
	Gestion de la limitation	Sélection conditionnelle dynamique dans des ensembles limites	
	Traitement de données avec descripteur temps	Collecte, mémorisation et présentation de données avec descripteur temps dans un affichage tableau/tendance	
	Collecte des données de perturbation	Enregistrement de séquences d'événements et autopsie	
	Statistiques d'équipement	Accumulation des opérations d'équipement	
	Planifications de commutation		
Traitement des alarmes		Traitement des alarmes et des événements	
Traitement de topologie		Analyse la connectivité de l'équipement et des états alimentés et construit des modèles de bus pour prendre en charge des fonctions réseau et présenter la connectivité sur l'affichage utilisateur. Peut comprendre une analyse de la connectivité de masse pour l'analyse d'anomalies.	Exploitant de transport, exploitant de zone de contrôle

Annex B
(informative)

Typical applications and functions

Table B.1 – Typical applications and functions

Application category			Actor (operator)
	Application name	**Comment/explanation**	
SCADA			Field operator
	Time synchronization	Synchronize RTU time tags	
	RTU and PLC communication	Field data acquisition using RP570, IEC 60870-5-101, …	Substation operator
	Remote center communication	Inter -site communications using ICCP, ELCOM, …	
	Signal processing	Filtering, scaling, engineering unit conversion, quality checking, limit checking	
	Data processing	Generalized calculations, status, analog, and accumulator value, quality processing	
	Tagging	Tagging is used to attach tags at arbitrary real world objects represented in a SCADA	
	Single line diagrams	Page based graphical process displays	
	Status lists	SCADA/EMS object status reports	
	Supervisory control	Device and sequence control (interlocked and non-interlocked)	
	System control		
	Intersite control		
	Limit management	Dynamic conditional selection on limit sets	
	Time tagged data processing	Collection, storage and presentation of time tagged data in tabular/trend displays	
	Disturbance data collection	Sequence of event recording and Post mortem review	
	Equipment statistics	Accumulation of equipment operations	
	Switching schedules		
Alarm Processing		Alarm and event processing	
Topology processing		Analyzes equipment connectivity and energized status and builds bus model to support network functions and for connectivity presentation on user displays. May include ground connectivity analysis for fault analysis.	Transmission operator, control area operator