SURFACE VEHICLE STANDARD

J2293-2	FEB2014

Issued 1997-06
Stabilized 2014-02

Superseding J2293-2 JUL2008

Energy Transfer System for Electric Vehicles - Part 2: Communication Requirements and Network Architecture

RATIONALE

This stabilized Recommended Practice documents for reference the historical state of energy transfer systems and communications for electric vehicles as they existed in 2008, as defined in SAE J1772 (per published version 11-1-2001) for conductive charging and SAE J1773 (per published version 11-1-1999) for inductive charging.

SAE J1772 continues to be updated to reflect the latest in conductive charging technology. See the latest available version of J1772.

SAE J1773 remains unchanged for inductive charging.

Documentation for the now-emerging “wireless” inductive charging systems will be published when available.

Grid power quality for supplying charging systems is covered in SAE document series J2894.

For state-of-the-art documentation on charging communications, refer to the SAE documents in the series J2836, J2847, J2931, and J2953.

STABILIZED NOTICE

This document has been declared "Stabilized" by the SAE Hybrid - EV Committee and will no longer be subjected to periodic reviews for currency. Users are responsible for verifying references and continued suitability of technical requirements. Newer technology may exist.

TO PLACE A DOCUMENT ORDER: **Tel: 877-606-7323 (inside USA and Canada)**
Tel: +1 724-776-4970 (outside USA)
Fax: 724-776-0790
Email: CustomerService@sae.org
SAE WEB ADDRESS: **http://www.sae.org**

SAE values your input. To provide feedback on this Technical Report, please visit
http://www.sae.org/technical/standards/J2293/2_201402

FOREWORD

This SAE Recommended Practice is intended as a guide toward standard practice and is subject to change to keep pace with experience and technical advances.

The ability of Electric Vehicles (EVs) to correctly operate with the off-board electrical supply equipment used to charge them is a major concern for the producers and consumers of such vehicles. This concern stems from the cost of developing an electric “fueling” infrastructure. If EVs are to gain wide acceptance, this infrastructure must, at the least, provide consumers with the same level of convenience that is available for today’s fossil-fueled vehicles. Ideally, there would be a single set of physical and functional requirements for the interaction between an EV and the off-board equipment. Designing to those requirements would allow any EV to charge with any off-board equipment, regardless of the producer of the vehicle or off-board equipment. This would be similar to today’s internal combustion engine vehicles and gasoline pumps for unleaded fuels.

Today, there are two electrical energy coupling methods under consideration for EVs. One *conductively* transfers electrical energy (AC or DC) through metallic contacts. The other *inductively* transfers electrical energy through a separable transformer. These couplings are detailed in SAE J1772 and SAE J1773, respectively. Each of these physical coupling methods includes a means to communicate data that are necessary to control the transfer of energy to the vehicle. Unfortunately, these coupling methods are not physically compatible. Therefore, interoperability between EVs and equipment using different methods is not possible.

While interoperability between conductively and inductively coupled systems is not physically possible, each coupling method performs the same basic functions. When the combination of the EV and the off-board equipment is considered as a complete energy transfer system, there is no reason for the system's functional requirements to differ due to the physical coupling method. The only variation as a result of coupling method is the location (on-board or off-board) where specific functions are accomplished. Defining standards for certain functions of the total system and standards for the communication of data that pass between the EV and the off-board equipment will insure interoperability of equipment with common coupling methods. Different coupling methods will require a different system architecture, not different functional requirements.

Control of the charging of an EVs storage battery is specific to the type of battery and the configuration of the vehicle. Also, energy may be brought on-board a vehicle for purposes other than charging of the battery. These other purposes are specific to the needs of a particular vehicle. From these facts, it follows that the EV should be in control of the transfer of energy from the off-board equipment. This way, EVs can have significantly different needs without forcing functional differences into the off-board equipment. It also establishes the EV (and its producer) as responsible to properly charge batteries, and allows charging requirements to evolve with developing battery technologies and EV experience.

This document will define a set of functional requirements for an Energy Transfer System (ETS) for Electric Vehicles, independent of energy coupling method. It will also serve as an "umbrella" document by reference of other SAE documents written or modified to accommodate this application. It will define three different physical system architectures that correspond to:

a. Conductive AC coupling

b. Inductive coupling

c. Conductive DC coupling

Requirements will be included and detailed only to the level that will insure functional interoperability for systems with common physical architectures. When designing systems, there will be additional requirements to consider that do not affect interoperability and are not included here. If requirements are found that affect functional interoperability, they should be considered for subsequent inclusion under the SAE J2293 umbrella.

This document has been jointly developed by the Electric Power Research Institute - National Electric Vehicle Infrastructure Working Council (EPRI - IWC), Data Interface Committee and the SAE Electric Vehicle Charging Controls Task Force. The efforts of all who participated are greatly appreciated.

TABLE OF CONTENTS

1. SCOPE

SAE J2293 establishes requirements for Electric Vehicles (EV) and the off-board Electric Vehicle Supply Equipment (EVSE) used to transfer electrical energy to an EV from an Electric Utility Power System (Utility) in North America. This document defines, either directly or by reference, all characteristics of the total EV Energy Transfer System (EV-ETS) necessary to insure the functional interoperability of an EV and EVSE of the same physical system architecture. The ETS, regardless of architecture, is responsible for the conversion of AC electrical energy into DC electrical energy that can be used to charge the Storage Battery of an EV, as shown in Figure 1.

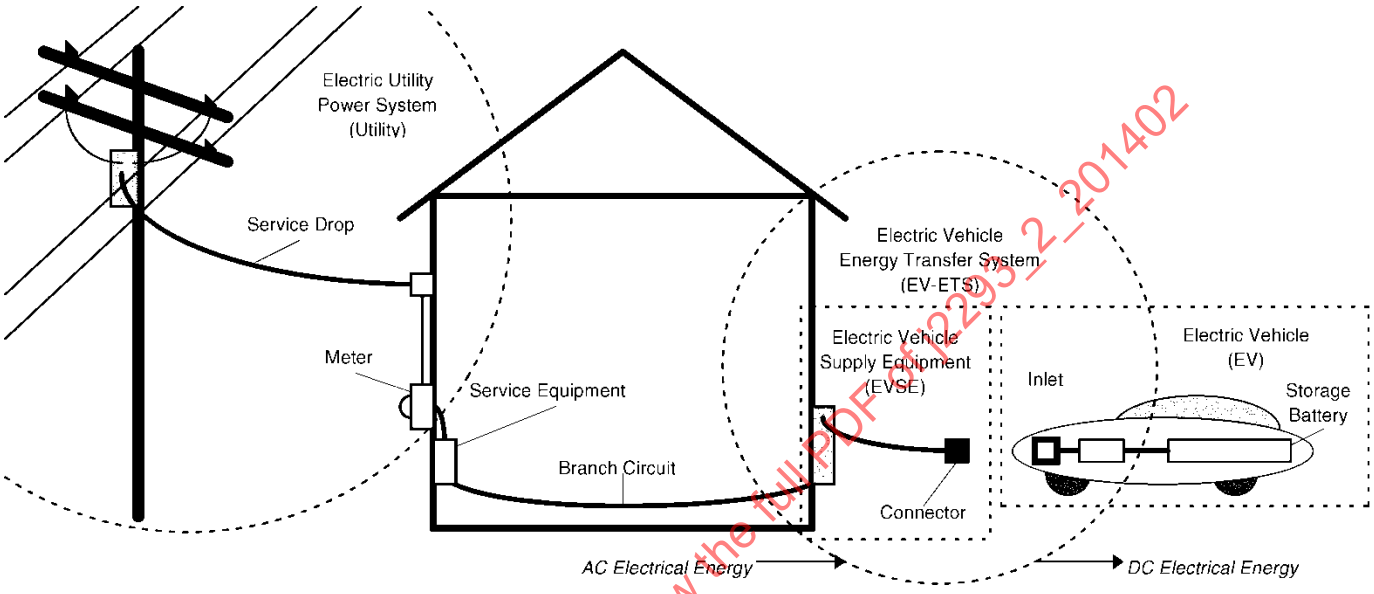

FIGURE 1 - ELECTRIC VEHICLE ENERGY TRANSFER SYSTEM PHYSICAL CONTEXT

The different physical ETS system architectures are identified by the form of the energy that is transferred between the EV and the EVSE, as shown in Figure 2. It is possible for an EV and EVSE to support more than one architecture.

This document does not contain all requirements related to EV energy transfer, as there are many aspects of an EV and EVSE that do not affect their interoperability. Specifically, this document does not deal with the characteristics of the interface between the EVSE and the Utility, except to acknowledge the Utility as the source of energy to be transferred to the EV.

The functional requirements for the ETS are described using a *functional decomposition* method. This is where requirements are successively broken down into simpler requirements and the relationships between requirements are captured in a graphic form. The requirements are written as the transformation of inputs into outputs, resulting in a model of the total system.

Each lowest level requirement is then allocated to one of four functional groups (FG) shown in Figure 2. These groups illustrate the variations of the three different system architectures, as the functions they represent will be accomplished either on an EV or within the EVSE, depending on the architecture. Physical requirements for the channels used to transfer the power and communicate information between the EV and the EVSE are then defined as a function of architecture. System architecture variations are referred to as follows:

a. Type A—Conductive AC System Architecture—J2293-1—6.2.1

b. Type B—Inductive System Architecture—J2293-1—6.2.2

c. Type C—Conductive DC System Architecture—J2293—6.2.3

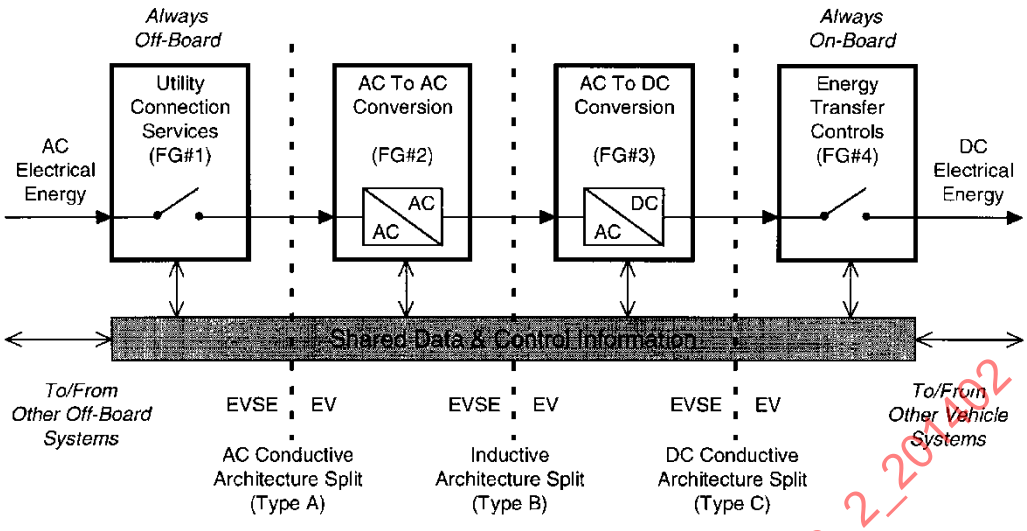

FIGURE 2 - EV ETS FUNCTIONAL GROUP SERIES ENERGY CONVERSION

The requirements model in Section 6 is not intended to dictate a specific design or physical implementation, but rather to provide a functional description of the system's expected operational results. These results can be compared against the operation of any specific design. Validation against this document is only appropriate at the physical boundary between the EVSE and EV. See Section 8.

1.1 Document Overview

This document consists of two parts, as shown in Figure 3. Each part is published separately. SAE J2293-1 describes the total EV-ETS and allocates requirements to the EV or EVSE for the various system architectures. It requires an SAE J1850-compliant network for communicating data and control information between an EV and EVSE.

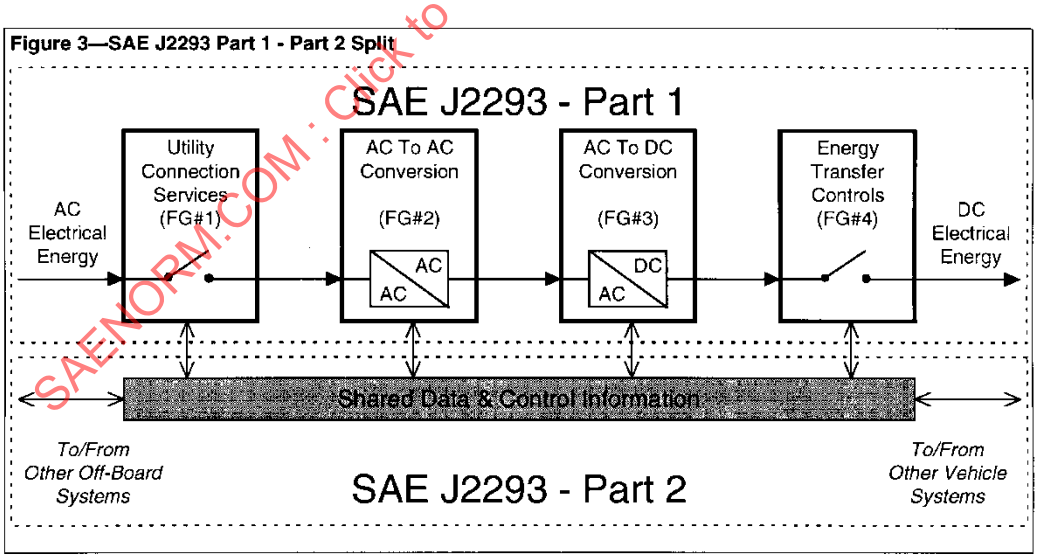

FIGURE 3 - SAE J2293 PART 1 - PART 2 SPLIT

SAE J2293-2 describes the SAE J1850-compliant communication network between the EV and EVSE for this application (ETS Network). It treats the network as a system with the EV and EVSE from Part 1 as external elements using the network. Each part has the following outline:

Section 1 Scope—Defines the purpose and content of the document.

Section 2 Reference Section

Section 3 Terms, Definitions—Defines Terms that are specific to the subject of this document and are not defined in another reference.

Section 4 Abbreviations, and Acronyms—Defines Abbreviations and Acronyms that are specific to the subject of this document and are not defined in another reference.

Section 5 System Definition and Context—Defines the purpose and context of the system by describing the relationship of the system to its environment (external elements). This section also defines specific constraints that will shape the functional and physical requirements.

Section 6 Functional System Requirements—Defines the specific requirements of the system using the functional decomposition method. The major functions are decomposed into simple, interrelated elements, without regard for whether a function will be implemented off-board or on-board the EV. These elements form a model of the total requirements.

Section 7 System Architecture—Defines the architecture(s) of the system. Each of the elements of the model in Section 6 are assigned to an EV or the EVSE. In Part 1, elements are first grouped into four functional groups. These groups form a series chain of energy conversions, as shown in Figure 2, that is representative of any of the ETS architecture.

Section 8 Validation—Identifies which of the requirements in Section 6 shall be validated for each of the architectures described in Section 7. Specific test procedures are defined as required.

Section 9 Data Dictionary—Defines each flow used as a part of the requirements model in Section 6. These are the minimum requirements that shall be considered when the system is actually implemented.

1.2 SAE Document Interrelationships

Figure 4 shows the interrelationships of SAE J1772, SAE J1773, SAE J1850, SAE J2178, and this document. SAE J2293 references these documents and provides additional information to form a complete set of requirements. There are other documents that will have an influence on the design of EV energy transfer system equipment. Where any conflicts exist in the following documents, the documents shall have the following order of precedence:

a. SAE J2293

b. SAE J2178

c. SAE J1850

d. SAE J1772

e. SAE J1773

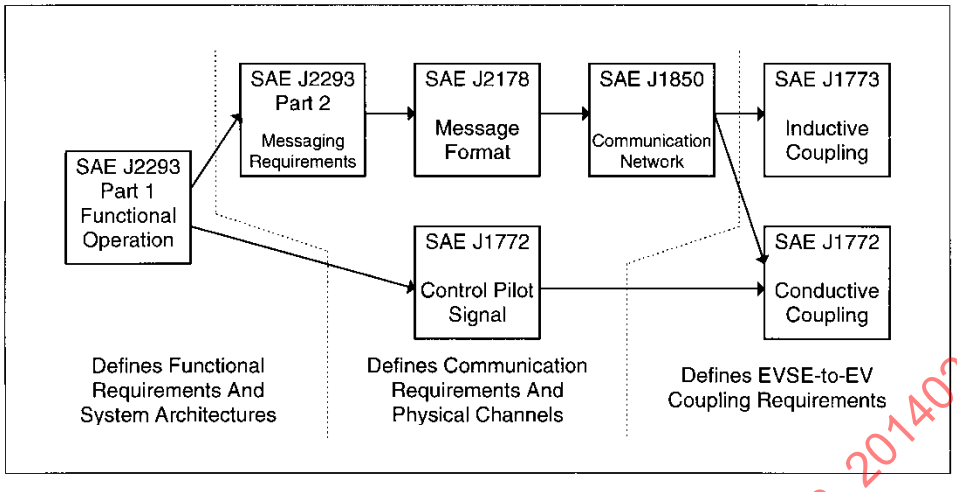

FIGURE 4 - SAE J2293 DOCUMENT INTERRELATIONSHIP

1.3 System Classification

System architecture variations shall be referred to as the following:

Type A—Conductive AC System Architecture
Type B—Inductive System Architecture
Type C—Conductive DC System Architecture

Optional content for each system type is identified separately. See J2293-1, Section 6, System Architecture.

2. REFERENCES

2.1 Applicable Publications

The following publications form a part of the specification to the extent specified herein. Unless otherwise indicated, the latest revision of SAE publications shall apply.

2.1.1 SAE Publications

Available from SAE International, 400 Commonwealth Drive, Warrendale, PA 15096-0001, Tel: 877-606-7323 (inside USA and Canada) or 724-776-4970 (outside USA), www.sae.org.

SAE J1715 Electric Vehicle Terminology

SAE J1772 SAE Electric Vehicle Conductive Charge Coupler

SAE J1773 SAE Electric Vehicle Inductively Coupled Charging

SAE J1798 Recommended Practice for Performance Rating of Electric Vehicle Battery Modules

SAE J1850 Class B Data Communications Network Interface

SAE J2178-1 Class B Data Communication Network Messages—Detailed Header Formats and Physical Address Assignments

SAE J2178-2 Class B Data Communication Network Messages—Part 2: Data Parameter Definitions

SAE J2178-3 Class B Data Communication Network Messages—Part 3: Frame IDs for Single-Byte Forms of Headers

SAE J2178-4 Class B Data Communication Network Messages—Message Definitions for Three Byte Headers

2.2 Related Publications

The following publications are provided for information purposes only and are not a required part of this document.

2.2.1 NFPA Document

Available from the National Fire Protection Agency, 1 Batterymarch Park, Quincy, MA 02169-7471, Tel: 617-770-3000, www.nfpa.org.

NFPA-70-1996 National Electric Code (NEC)® - Article 625

2.2.2 Other Publication

Available from Dorset House Publishing, 353 West 12th Street, New York, NY 10014, Tel: 1-800-DH-BOOKS, www.dorsethouse.com.

Strategies for Real-Time System Specification; Derek J. Hatley and Imtiaz A. Pirbhai, 1988

3. DEFINITIONS

3.1 Battery

See Electric Vehicle Storage Battery.

3.2 Branch Circuit

The circuit conductors between the final overcurrent device protecting the circuit and the equipment supplied by the circuit. It is typically an unswitched circuit from the service equipment (fuse box) to an appliance. For this application, the appliance is the Electric Vehicle Supply Equipment (EVSE).

3.3 Control Flow

The representation of information, energy, or matter that is used to alter the behavior of a system. These are shown as dotted arrows on a data flow diagram (DFD) to indicate the source and destination of the flow. A flow may be both a data and a control flow, depending on how it is used by the system. This construct is used as a part of the functional decomposition process.

3.4 Control Specification (C-spec)

The description of a combinational or sequential logic operation, the results of which can be used as the input to a process specification (P-spec), or to describe when a process is to be operational. Types include decision tables (DT), state transition diagrams (STD), and process activation tables (PAT). C-specs are shown as a single vertical bar on a data flow diagram (DFD) and are referenced by the DFD level they appear on, followed by an s-index (e.g. 2.3.3-s1, 4-s7). This construct is used as a part of the functional decomposition process.

3.5 Data Flow

The representation of information, energy, or physical matter that is transformed by a system. These are shown as solid arrows on a Data Flow Diagram (DFD) to indicate the source and destination of the flow. A flow may be both a data and a control flow, depending on how it is used by the system. This construct is used as a part of the Functional Decomposition process.

3.6 Data Flow Diagram (DFD)

A diagram that shows the relationship between portions of a larger, more complex process as the process is decomposed. It captures the flow of data, energy, or physical matter between the portions. This construct is used as a part of the Functional Decomposition process.

3.7 Decision Table (DT)

A type of C-spec that defines the conditions required to determine the value of a combinational logic function. A combinational logic function, as opposed to a sequential logic function, is one where its present value (output) is dependent only on the present value of its arguments (inputs). See 3.23.

3.8 Electric Utility Power System (Utility)

The system that generates and delivers commercial electrical power to a residential or commercial building or facility. It extends to and includes a billing apparatus (electric meter).

3.9 Electric Utility/Local Load Management System (LMS)

A system that is responsible to monitor and control the load on some portion of the Utility or local premises' feeder and branch circuits. The goal of control may be to prevent overload or to reduce the cost of energy based on a specific billing agreement.

3.10 Electric Vehicle Storage Battery (Battery)

A group of electrochemical cells electrically connected in a series and/or parallel arrangement, the principal purpose of which is to provide DC electrical energy to propel the EV.

3.11 Electric Vehicle Supply Equipment (EVSE)

The equipment from the branch circuit to, and including, the connector that couples to the electric vehicle inlet, the purpose of which is to transfer electric energy to an EV. This equipment is located off-board the vehicle.

3.12 Elementary Process

A process in a data flow diagram (DFD) that is not decomposed further. Its function or activity is described by a process specification (P-spec). This construct is used as a part of the Functional Decomposition process.

3.13 Energy Transfer

The process of flowing energy to the EV from the EVSE.

3.14 Energy Transfer Strategy

A strategy that accounts for all of the electrical energy needs of an EV and the present status of all on-board equipment, including the EV Storage Battery. It determines the rate that energy is to be transferred to the EV and how the ETS shall be operated to accomplish this.

3.15 Energy Transfer System (ETS)

A system that is distributed between an Electric Vehicle (EV) and the off-board Electric Vehicle Supply Equipment (EVSE), the purpose of which is to transfer electrical energy from the Utility Power System (Utility) to the EV Storage Battery and other vehicle loads. The EV and EVSE must be connected together for energy to be transferred.

3.16 Functional Decomposition

A method used to describe the functional and behavioral requirements of a system, component, or device, captured as a model of the requirements. It involves the hierarchical break-down of complex requirements into simple, easy to describe pieces. The flow of information, energy, and matter between the pieces is captured, along with the process and control requirements.

3.17 Interoperability

The condition where components of a system, relative to each other, are able to work together to perform the intended operation of the total system. As an example, a 10 mm box-end hand wrench and a 10 mm socket wrench are interoperable, relative to a 10 mm hex-head bolt. The wrench and the bolt are both parts of a fastening system. The fact that the system will perform as required with either wrench establishes the interoperability of the wrenches and the bolt.

3.18 Off-Board/On-Board Boundary

The point where the ETS is divided into two physical parts. One part becomes realized within the off-board Electric Vehicle Supply Equipment (EVSE). The other part becomes realized within an Electric Vehicle. This boundary will be in different places, depending on the system architecture.

3.19 Power Stage

A physical and/or logical section of power conversion equipment that can provide conversion over a portion of the total conversion power range of the equipment. The range of a specific power stage will overlap with its adjacent stages.

3.20 Process

A function or activity that can be described as an input/output relationship or as the combination of simpler processes. A process, a basic building block of a data flow diagram (DFD).

3.21 Process Activation Table (PAT)

A type of C-spec that defines the exact conditions required for the indicated P-specs to be active. This construct is used as a part of the Functional Decomposition process.

3.22 Process Specification (P-spec)

The description of an elementary process. The description may include text, figures, and tables. The description captures the input/output relationship for all flows shown on its related data flow diagram (DFD). A P-spec is shown as a circle on a DFD, and has no further “child” processes below it. This construct is used as a part of the Functional Decomposition process.

3.23 State Transition Diagram (STD)

A type of C-spec that defines the exact conditions required to determine the value of a sequential logic function. A sequential logic function, as opposed to a combinational logic function, is one where its present value (output) is dependent on the present value of its arguments (inputs) and some number of the previous values of the arguments (previous inputs). See 3.7.

3.24 Type A Architecture

A form of the ETS where AC electrical energy is transferred from the EVSE to the EV, across the off-board/on-board boundary, via a conductive coupling.

3.25 Type B Architecture

A form of the ETS where AC electrical energy is transferred from the EVSE to the EV, across the off-board/on-board boundary, via an inductive coupling.

3.26 Type C Architecture

A form of the ETS where DC electrical energy is transferred from the EVSE to the EV, across the off-board/on-board boundary, via a conductive coupling.

3.27 Utility

See Electric Utility Power System.

4. ABBREVIATIONS AND ACRONYMS

C-spec Control Specification
CSMA Carrier Sense Multiple Access
DFD Data Flow Diagram
DT Decision Table
EPRI Electric Power Research Institute
ETS Energy Transfer System
EV Electric Vehicle
EV-ETS Electric Vehicle Energy Transfer System
EVSE Electric Vehicle Supply Equipment
FG Functional Group
FG#m Functional Group m
HVAC Heating, Ventilation, Air Conditioning
IWC Infrastructure Working Council
LMS Load Management System
NEC® National Electric Code®
P-spec Process Specification
PAT Process Activation Table
SAE SAE International
STD State Transition Diagram
VS Ventilation System

5. SYSTEM DEFINITION AND CONTEXT

5.1 System Overview

When an EV and an EVSE are coupled, energy and information is exchanged. SAE J2293-1 describes the overall information and energy exchange that occurs. SAE J2293-2 describes the requirements for exchanging data between an EV and an EVSE via a SAE J1850 network. The system that exchanges this data is the Energy Transfer System (ETS) Network. Together SAE J2293-1 and SAE J2293-2 define the requirements for communication between an EVSE and EV.

SAE J2293-1 defines three different ETS architectures: AC Conductive, Inductive, and DC Conductive. SAE J2293-1 defines the information that is passed between an EV and EVSE. The different architectures require different information to be passed.

SAE J2293-2 defines the requirements for a common set of methods to exchange information between the EV and the EVSE regardless of the architecture. SAE J2293-2 defines the relationship between the data flows defined in SAE J2293-1 and the messages that communicate those data flows.

Figure 5 shows how the ETS Network interacts with the elements of the ETS to perform its function. This diagram is a variation of Figure 3. It is expanded to illustrate the role that the ETS network provides in the Energy Transfer System.

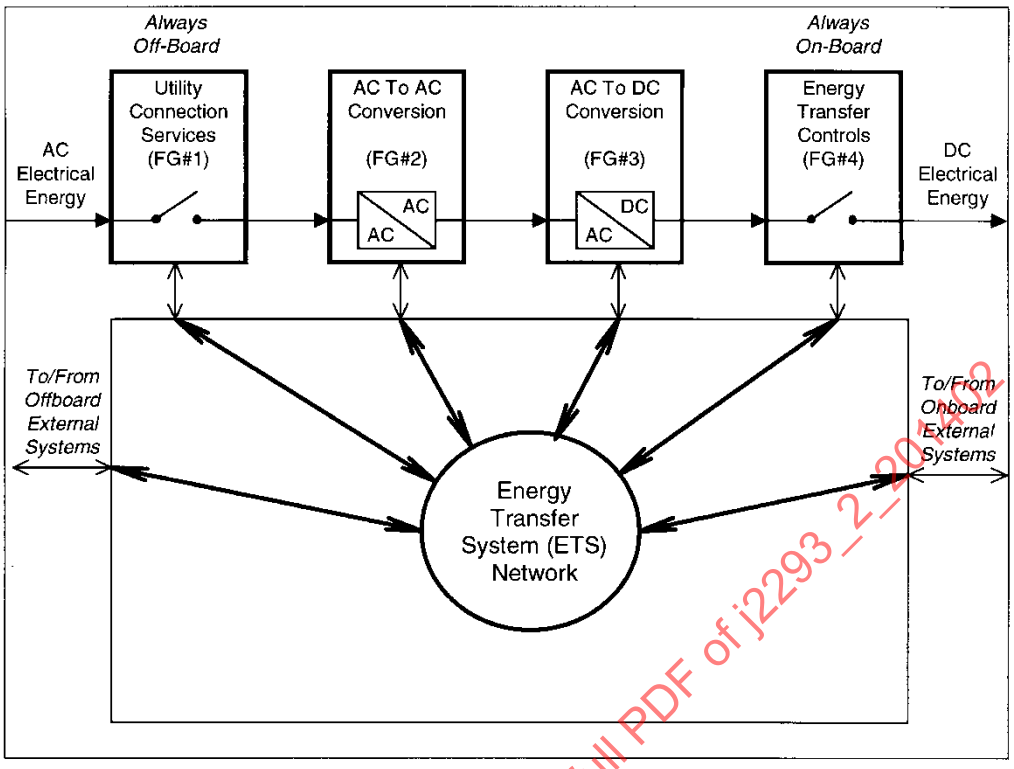

FIGURE 5 - ILLUSTRATION OF ETS NETWORK CONTEXT

SAE J2293-2 defines the ETS Network so it is compatible with existing SAE J1850 requirements and implementations. Features of this compatibility include:

a. The ETS Network accommodates operation with other vehicle applications and off-board applications without the need for vehicle network gateways. However this does not preclude implementation of a vehicle gateway should the vehicle manufacturer decide to do so.

b. The protocols allow vehicles built with SAE J1850 implementation type 1 and implementation type 2 networks to communicate with EVSEs using the ETS Network.

c. These protocols allow a vehicle using unconsolidated headers or consolidated headers to communicate with the ETS network.

d. These protocols accommodate the addition of applications beyond the scope SAE J2293-1. These additional applications can either be open standards or manufacturer specific. Both types of applications can co-exist on this network. This permits manufacturers to meet needs in the market place while maintaining compatibility with existing standards and either EVSEs or vehicles built to those standards.

To illustrate the scope of this document, Figure 6 shows a possible node in the ETS network. This node consists of several parts.

a. External Interfaces

b. Application Functions

c. Application Network Moding Functions

d. Network Communication Functions

e. Network Electrical Interfaces

Depending on the location and the applications in a node, the applicability of SAE J2293-2 changes. Table 1 identifies how the requirements apply. Figure 7 provides an example of this document's applicability to different nodes in the ETS network.

TABLE 1 - SAE J2293-2 REQUIREMENT'S SCOPE

	J2293-1 Application (ETS Application)	Any Other Off-Board Application which uses the ETS Network	Any Other On-Board Application which uses the ETS Network
External Interfaces	Defined in SAE J2293-1	Not Defined	Not Defined
Application Functions	Defined in SAE J2293-1	Not Defined	Not Defined
Application Network Moding Functions	Defined	Partially Defined	Partially Defined
Network Communication Functions	Defined	Partially Defined	Partially Defined
Network Electrical Functions	Defined	Defined	Defined

The ETS Network constrains the portion of the EVs on-board network that connects to the EVSE. These constraints only apply while the EV is connected,

a. The EV portion of the ETS network must either be SAE J1850 implementation type 1 or 2

b. The EV portion of the ETS network must use messaging which is compatible with either consolidated or unconsolidated headers

c. The EV portion of the ETS network must insure that enough network bandwidth is available for ETS communication so that messages meet their latency goals.

5.2 Overview of Requirements Organization

These requirements provide a complete description of the ETS Network. The requirements are grouped into several categories:

a. Definition of Interfacing Systems and Applicable Interfaces

b. Definition of Physical and Functional Requirements

c. Allocation of SAE J2293-1 Requirements to the ETS Network

d. Allocation of Requirements to Elements and Interfaces of the ETS Network

e. Data Dictionaries

5.2.1 Definition of Interfacing Systems and Applicable Interfaces

Any system performs its function by interacting with the world beyond itself. The ETS Network interacts with several systems. These external systems provide inputs and receive outputs from the ETS Network.

Paragraph 5.3 identifies and defines these systems. The interfacing systems are described. Any recommendations, expectations, or requirements on these systems' functions are stated.

Paragraph 5.4 defines the interfaces between the ETS Network and these systems. The interface is described. Recommendations, expectations, and requirements for each interface are stated.

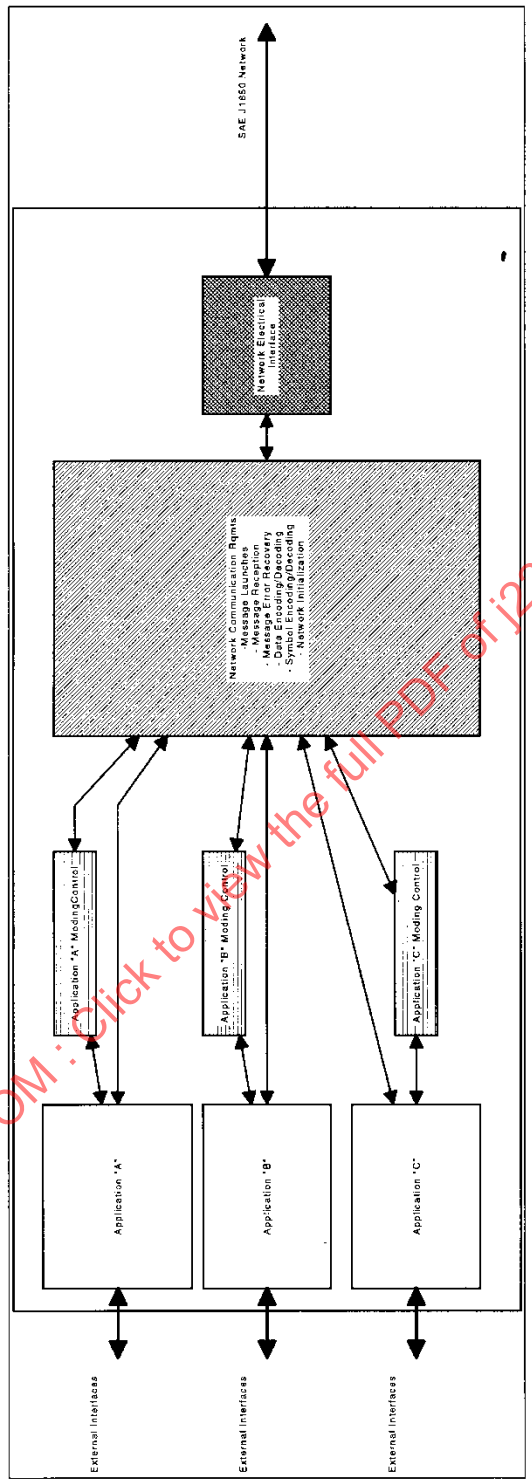

FIGURE 6 - POSSIBLE SAE J1850 NODE IN ETS NETWORK

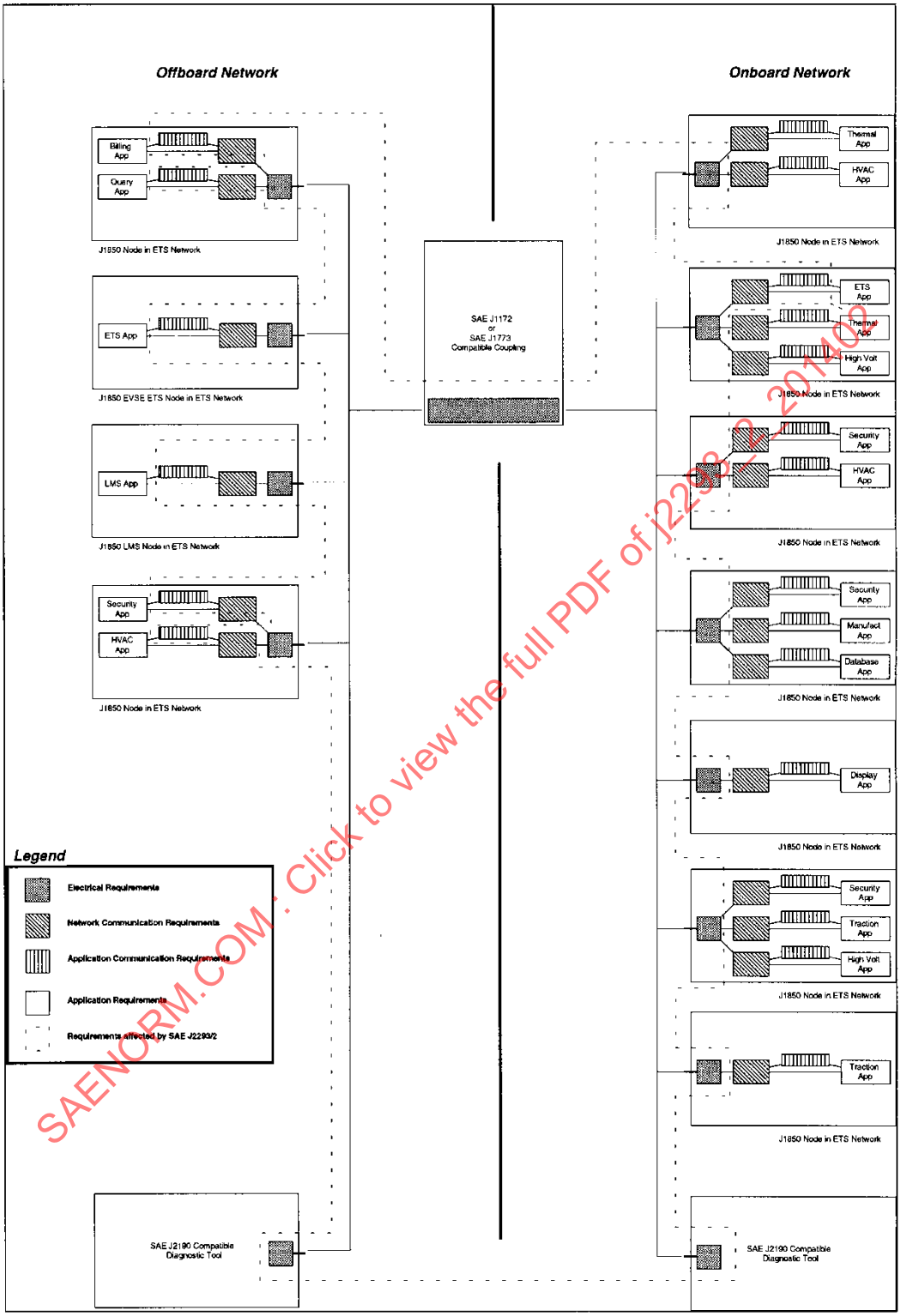

FIGURE 7 - POSSIBLE ETS NETWORK CONFIGURATION

5.2.2 Definition of Physical and Functional Requirements

The ETS network transports information from one physical location in the Energy Transfer System to another. To achieve this function, detailed requirements are defined for the ETS Network. Some of these requirements are physical in nature. These requirements define how some aspect of the system must be built. Other requirements are functional in nature. These requirements define how an aspect of the system must behave. All requirements are decomposed to the point that individual requirements can be wholly allocated to individual elements of the ETS.

The general guidelines for the development of the requirements are documented in sections:

a. 5.5 Functional Content

b. 5.6 Functional Constraints

c. 5.7 Physical Content

d. 5.8 Physical Constraints

Section 6 defines the detailed requirements. Unique requirements are defined in sections:

a. 6.1 Transceive Off-Board Information

b. 6.2 Connect ETS Networks

c. 6.3 Transceive EV Information

These requirements reference templates, which are groups of recurring functional requirement. These templates allow the same basic requirements to be applied in a broad range of variations, while simplifying the documentation. These templates are defined in sections:

a. 6.5 Manage Application Slave Node Communication Template

b. 6.6 Manage Application Master Node Communication Template

c. 6.7 Detect Application Transmitter Fault Template

d. 6.8 Launch Message Template

e. 6.9 Maintain Parameter Template

5.2.3 Allocation of SAE J2293-1 Requirements to the ETS Network

SAE J2293-1 allocates requirements to the ETS Network. These allocated requirements are found in sections:

a. 5.6 Functional Constraints

b. 5.8 Physical Constraints

c. 6.10 ETS Network Message Requirements

The tables in 6.10 are generated from the data dictionary in SAE J2293-1.

5.2.4 Allocation of Requirements to Elements and Interfaces of the ETS Network

The requirements in this document are allocated to elements of the ETS network. Those elements are identified in Section 7. Table 57 defines which requirements are allocated to which elements of the ETS Network.

5.2.5 Data Dictionaries

To define the content of the information, matter, or energy that is exchanged between the processes in the requirements model, the data dictionary is provided. Section 9 provides this definition for the flows between the processes in 6.1, 6.2, and 6.3.

The templates in sections:

a. 6.5 Manage Application Slave Node Communication Template

b. 6.6 Manage Application Master Node Communication Template

c. 6.7 Detect Application Transmitter Fault Template

d. 6.8 Launch Message Template

e. 6.9 Maintain Parameter Template

Each have their own data dictionary. These data dictionaries are contained within the template. The data dictionaries are kept separate from the data dictionary for the ETS Network because the template is used multiple times within the system.

5.3 System Context

Figure 8 illustrates the interfaces and the system context for the ETS Network.

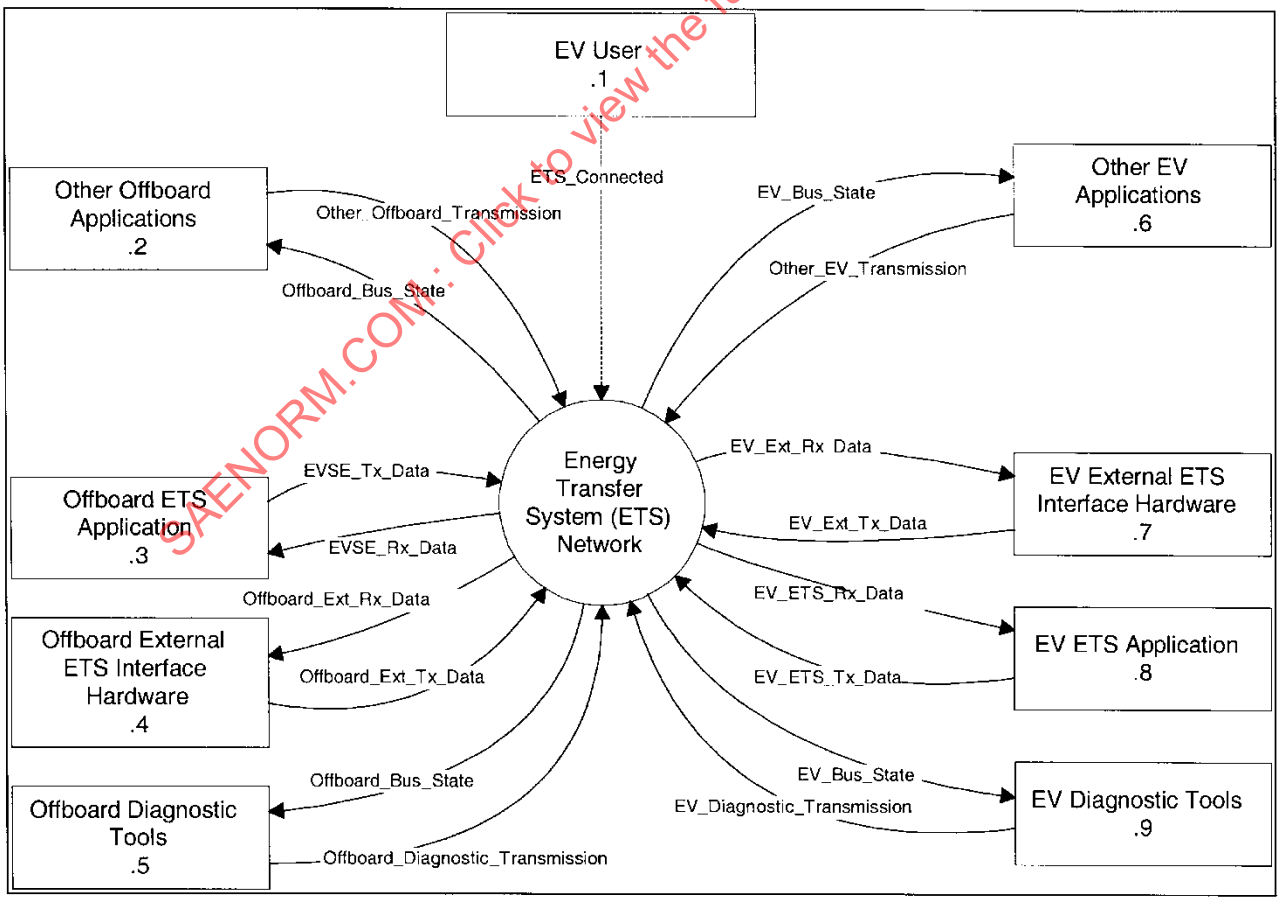

FIGURE 8 - ETS NETWORK FUNCTIONAL CONTEXT DIAGRAM

The systems that interface to the ETS Network are described in the sections that follow.

5.3.1 EV User

The EV User interacts with the network by physically connecting the EV and the EVSE. When this occurs, the EV network and the off-board network combine to become the ETS network.

5.3.2 Other Off-Board Applications

The Other Off-board Applications are any processes that are located off-board and use the ETS network. These processes may communicate internal to the EVSE or share data with the EV. Examples include security, billing, thermal, and entertainment applications.

While connected to an EV, the Other Off-board Applications will not use the ETS network unless specifically enabled by the EV. Applications which are enabled for communication are referred to as "Application Slaves" in this document. This refers to the concept that these applications are slaved to the commands from an "Application Master." All off-board applications are "Application Slaves." Paragraph 6.5 describes these methods.

5.3.3 Off-Board ETS Application

The Off-board ETS Application is the group of processes in the EVSE specified in SAE J2293-1 and allocated to the EVSE based on the ETS architecture used.

These off-board processes exchange data with processes on the EV to control the energy transfer from the Electric Power Utility System into the EV.

5.3.4 Off-Board External ETS Interface Hardware

The Off-board External ETS Interface Hardware is optional content that communicates information sourced or received externally as defined in SAE J2293-1 data dictionary. This hardware may include displays, switches, a Load Management System, and other user interfaces. The device that generates the "Preference Override" is an example of this interface hardware.

5.3.5 Off-Board Diagnostic Tools

Diagnostic Tools are devices which:

a. Generate interrogation messages for nodes on a network for the purpose of diagnosing the health of a system and identify sources of problems

b. Monitor messages on a network to diagnose the health of a system and identify sources of problems

EV and EVSE manufacturers may provide for access of diagnostic tools to the ETS Network. These diagnostic tools should meet the following requirements to insure proper operation of the network.

a. Off-board Diagnostic tools should interrogate and diagnose EV and off-board functions using SAE J2190 compatible messages. Definition of the messages and protocols used to communicate between diagnostic tools and the ETS Network are not defined in this document.

b. Off-board Diagnostic Tools should either plug into the ETS network or be integrated into the network.

c. Any EVSE/EV Diagnostic procedures which may prevent ETS messages from meeting latency goals identified in Table 56 should disable energy transfer prior to the start of the procedure.

d. Any EVSE service procedures which uses non-standard network modes or causes the network to change configuration should only be used when the EV and EVSE are not connected. When connected to an EV, manufacturer specific diagnostic messages should not be used by EVSE Diagnostic Tools unless enabled by the EV. No definition of how this occurs is defined in this document.

5.3.6 Other EV Applications

The Other EV Applications are processes which reside in the EV and use the ETS network. These processes may communicate internal to the EV or with the EVSE.

The Other EV Applications may use the ETS network as necessary at the vehicle manufacturers discretion. These applications may only use the network in such a way that the ETS messages meet the latency goals defined in Table 57. Failure to meet these latency objectives may prevent the proper transfer of energy.

5.3.7 EV ETS Application

The EV ETS application is the group of processes in the EV which are specified in SAE J2293-1 and allocated to the EV based on the ETS architecture used.

These processes exchange data with processes in the EVSE to control the energy transfer from the utility into the EV.

5.3.8 EV External ETS Interface Hardware

The EV External ETS Interface Hardware is optionally provided to communicate information which is sourced or has a destination of external as defined in SAE J2293-1 data dictionary. This hardware may include displays, switches, the vehicle inlet, and other user interfaces. The device which generates the "Preference Override" is an example of this interface hardware.

5.3.9 EV Diagnostic Tools

The requirements for EV Diagnostics Tools are identical to the requirements for EVSE Diagnostic Tools with the exception that any EV service procedures which use non-standard bus modes or cause the network to change configuration should disable the EVSE communication before the start of the procedure.

5.4 External Interfaces

The ETS Network interfaces with external systems. These interfaces are described to the detail sufficient to establish data flow requirements and reference to interface specifications. Unless an interface specification is defined, any implementation which communicates the required data at the necessary rate and resolution is sufficient.

5.4.1 EV User Interface

This is the mechanical interface presented by the coupling for the EV user to connect the EV to the EVSE.

5.4.2 Other Off-Board Application Interface

The ETS Network may be used to exchange between any nodes on the network. The interface between the ETS Network and Other Off-board Applications is the SAE J1850 network off-board the vehicle.

a. Any off-board application which uses the ETS Network shall follow the conventions established in 6.5 for enabling transmission of messages, enabling reception of messages and generation of service requests to the vehicle.

Examples of off-board applications are security, billing, convenience, and remote diagnostic applications.

5.4.3 Off-Board ETS Application Interface

The ETS Network transports information from the EVSE ETS Application to the EV ETS Application. The interface between the ETS Network and the EVSE ETS Application may be implemented as any mechanism which can communicate the necessary data flows to the ETS Network while maintaining latency goals identified in Table 57. Possible implementations would include shared RAM where the EVSE ETS Application and the ETS Network can exchange information or a data bus between the EVSE ETS Application and the ETS Network.

5.4.4 Off-Board External ETS Interface Hardware Interface

The ETS Network allows for transport of information from the external user interfaces or Load Management System (LMS) to the ETS Application. The interface between the ETS Network and the Off-Board External ETS Interface Hardware or LMS may be implemented as any mechanism which can communicate the necessary data flows to the ETS Network while maintaining latency goals identified in Table 57. Possible implementations would include shared RAM or a data bus between the Off-Board External ETS Interface and the ETS Network (i.e., a gateway node).

5.4.5 Off-Board Diagnostic Tools Interface

EVSE manufacturers may provide for access to the ETS network via a SAE J2190 compatible diagnostic tool. This tool may be used to perform diagnostics on the EV and the EVSE. This interface will be to the SAE J1850 network from any location off-board.

a. All off-board diagnostic communication via the ETS network should be SAE J2190 message format compatible.

5.4.6 Other EV Application Interface

Other applications designed in the vehicle may use the ETS network. Any EV Application which shares this network should meet the following restrictions:

a. The messages used by Other EV Applications should be SAE J2178 consolidated header compatible or SAE J2178 unconsolidated header compatible.

b. The message bandwidth requirements should be designed to assure that the latency goals of messages on the ETS Network meet the timing requirements defined in Table 57.

c. The bandwidth utilization should be less then 90% average utilization over 2 seconds for the ETS application combined with Other EV Application message traffic under worst case steady-state operation. This restriction is based on engineering judgment of inherent limitations in most implementations of these protocols.

5.4.7 EV External ETS Interface Hardware Interface

The ETS Network allows for transport of information from the external user interfaces in the EV to the ETS Application. The interface between the ETS Network and the Off-board External ETS Interface Hardware may be implemented as any mechanism which can communicate the necessary dataflows to the ETS Network while maintaining latency goals identified in Table 57. Possible implementations would include shared RAM where the LMS and the ETS Network can exchange information or a data bus between the Off-board External ETS Interface and the ETS Network (i.e., a gateway node).

5.4.8 EV ETS Application Interface

The ETS Network transports information from the EV ETS Application to the EVSE ETS Application. The interface between the ETS Network and the EV ETS Application may be implemented as any mechanism which can communicate the necessary dataflows to the ETS Network while maintaining latency goals. Possible implementations would include shared RAM where the EV ETS Application and the ETS Network can exchange information or a data bus between the EV ETS Application and the ETS Network system (i.e., a gateway node).

5.4.9 EV Diagnostic Tool Interface

Requirements for this interface are identical to 5.4.5.

5.5 Functional Content

The EV ETS network provides the functions shown in Figure 9. Each process is decomposed to a level which can be simply described.

The functions performed by the network are organized into three categories

a. Transmission and reception of ETS information Off-board the EV.

b. Transmission and reception of ETS information in the EV.

c. Connection and interfacing of the EV and off-board networks.

5.6 Functional Constraints

The sections that follow document the higher level guidelines which contribute significantly to the functional requirements of the ETS network.

5.6.1 Provide Compatibility with Existing Hardware

These requirements provide compatibility, as much as possible, with the existing hardware which has been developed for J1850 and J2178 compatible networks.

5.6.2 Provide Compatibility with Existing SAE J2178 Messages and Strategies

To maintain compatibility with existing standards, all messages and strategies are compatible with SAE J2178 where possible.

5.6.3 Support for an Open Architecture

This document defines a protocol for communication between a vehicle and an off-board system. To allow maximum flexibility in the use of this network and its communications, it supports an “Open Architecture”. “Open Architecture” is used to describe a set of goals to enable future expansion of the network. The goals for “Open Architecture” of this system are:

a. Accommodate future growth of EV-EVSE shared applications (security, entertainment, remote preconditioning, etc.),

b. Accommodate Non-ETS applications operating on the vehicle (thermal management, entertainment, diagnostics, etc.) that use manufacturer specific protocols,

c. Accommodate evolution of this standard so that vehicle and chargers built to current the version are able to operate with later versions.

d. Allow EV and EVSE manufactures to provide applications that are not standardized without compromising the interoperability of EVSEs and EVs.

e. Allow compatibility with networks utilizing both three byte consolidated headers and single byte unconsolidated headers.

f. Failsoft if an improperly implemented system is connected to either an EV or an EVSE.

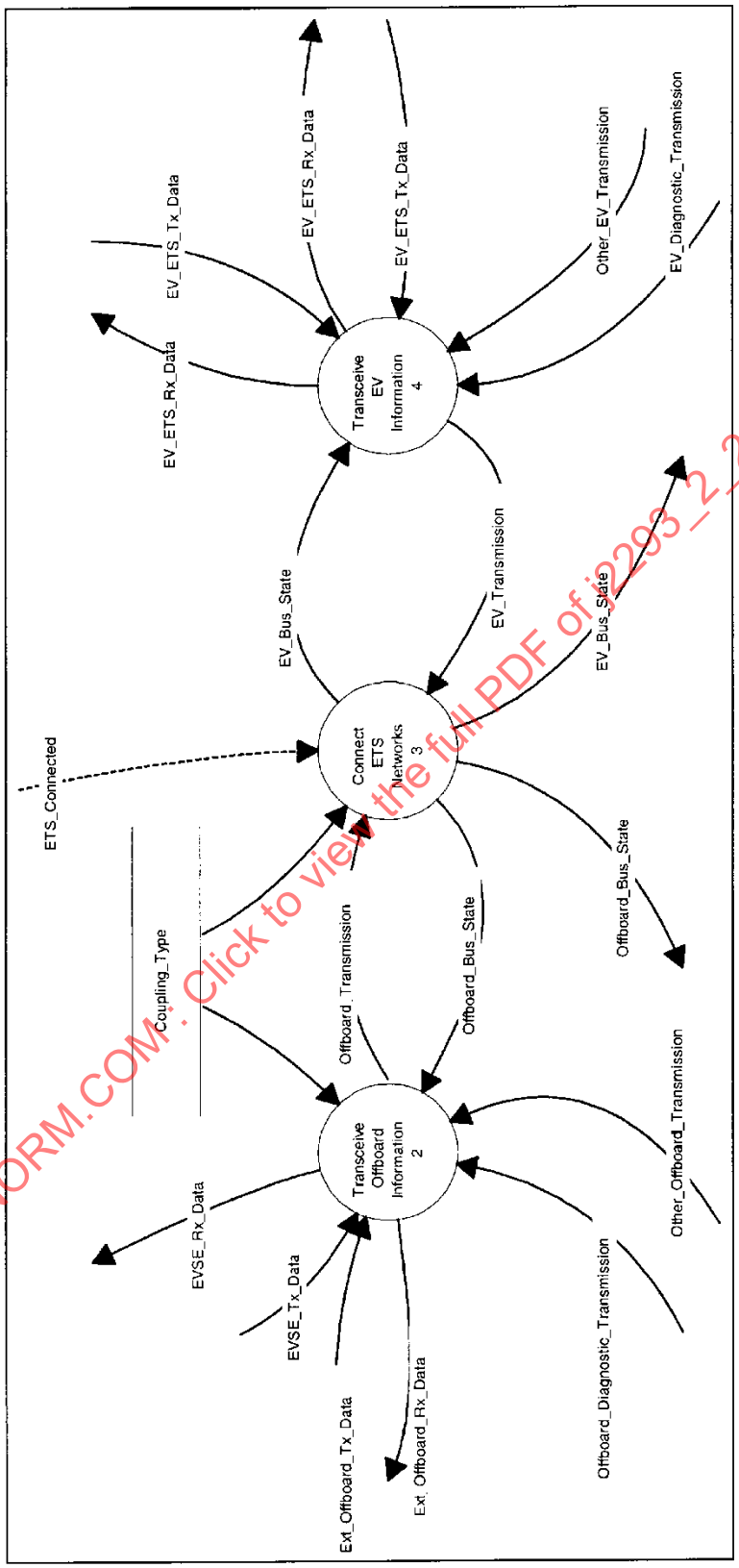

FIGURE 9 - ENERGY TRANSFER SYSTEM NEtWORK DFD

5.6.4 Minimize Network Bandwidth Utilization by ETS System

Maximum ETS Network utilization by the ETS application should be less than 10% average bus capacity over a period of 1 minute for continuous communications. ETS related initialization shall complete in less than 2 seconds with 100% of bus bandwidth available. These limits are to allow for sufficient bandwidth to

a. Facilitate future expansion of the functions over the EV-EVSE network

b. Reserve bandwidth to use by the EV manufacturer for use by other applications

5.6.5 Limit Maximum Message Transmission Rate

Per SAE J2293 Part 1 paragraph 4.5.5 Network Communication delay, the minimum interval between transmissions of any individual message for the ETS application shall be less than once per 100 msec.

5.6.6 Provide Protocol for identification of Non ETS Applications

For future expansion, the EV and the off-board applications will interact to enable functions that are not yet defined. This interaction can have an appreciable effect on the bus utilization and message latency. To constrain this bus utilization, a protocol is provided for off-board applications to identify themselves to vehicle applications with minimal bus impact.

5.6.7 The EV Uses Either Implementation 1 or Implementation 2 Media for the ETS Network

The EVSE supports both Type 1 SAE J1850 networks and Type 2 SAE J1850 networks. However, the EV shall only use one of these two networks. The use of both networks by a single EV for communication with an off-board system is not supported. The use of a network which does not conform to one of these types is not supported.

The Type 2 SAE J1850 bus should be able to operate in the degraded operating modes identified in SAE J1850.

5.6.8 ETS Nodes Issue a Network Wakeup to Initiate Communication

SAE J1850 nodes may enter a low power state when no communication is occurring on the network. In this low power state, processors may not be powered. These nodes may then wakeup and respond to communication when the network activity is first observed. To allow this behavior, a period of time between the first message on the network and the first useful information being transmitted is defined. This first message is sent following an inactive bus is defined.

5.6.9 Support a Subset of SAE J1850 Message Sequences

The ETS Network only supports a subset of the possible message operators and sequences defined in SAE J1850 and SAE J2178. This subset is used to provide the functionality required for the ETS subsystem, while simplifying and commonizing to existing communication strategies. Table 2 defines the operators supported for messaging. Table 3 describes the message sequences supported. The names for operators are commonized to simplify description of requirements. These operators can be referenced to SAE J2178 through use of Table 26 and Table 27.

TABLE 2 - SUPPORTED MESSAGE OPERATORS

Supported Message Operators
LD
RPT
REQ
RQCV

TABLE 3 - SUPPORTED MESSAGING SEQUENCES

Supported Sequences of Messages
LD w/o acknowledgment
RPT w/o acknowledgment
LD-> RPT
REQ -> RPT
RQCV -> LD

5.7 Physical Content

The physical content of a system that meets the system functional requirements may vary significantly. Section 7 describes the physical content of the ETS Network in detail.

5.8 Physical Constraints

The sections that follow document the higher level guidelines which contribute significantly to the requirements of the ETS network.

5.8.1 Utilize SAE J1850 Compatible Network

SAE J2293-1 specifies that the network used in the EV and the EVSE be compatible with a SAE J1850 Network. This requirement is stated in paragraphs:

a. EVSE-EV Communication

b. 4.5.2 EVSE-EV Communication

c. 6.2.1.3 Serial Data Interface

d. 6.2.2.2 Serial Data Interface

e. 6.2.3.2 Serial Data Interface

5.8.2 Node and Source Address Limitations

Since the SAE J1850 source addresses and electrical loading are allocated resources, the ETS system and its interfaces are allocated a limited amount of electrical loading and a limited quantity of SAE J1850 Source Addresses. The paragraphs which follow identify the general rules which govern these allocations.

Electrical Loading Allocations

a. The Off-Board Network has a Maximum of 8 SAE J1850 Standard Unit Loads.

b. The EV is allocated the balance of the remaining Standard Unit Loads available.

Source Address Allocations

a. The ETS Network has a maximum of 254 unique source addresses.

b. The ETS portion of the ETS Network has a Maximum of 4 SAE J1850 Nodes with unique Source Addresses. These four addresses are allocated to the combination of the EV and the EVSE depending on the architecture in use. The EVSE portion of the ETS Network has a Maximum of 3 SAE J1850 Nodes.

c. The EV portion of the ETS has a Maximum of 3 SAE J1850 Nodes with unique Source Addresses

d. The LMS portion of the ETS Network has a Maximum of 1 SAE J1850 Node with one unique Source Address. This source address is not defined in this document.

e. The off-board External ETS Interface Hardware has a maximum of one unique Source Address. This source address is not defined in this document.

5.8.3 Utilize SAE J1772 or SAE J1773 Compatible Couplers

SAE J2293-1 paragraph 5.5.1 EVSE EV Connection specifies that the physical coupling between the EV and the EVSE is either a SAE J1772 or SAE J1773 compatible coupler. SAE J1773 Couplers only support Type B ETS Architectures with Type 1 SAE J1850 Networks. SAE J1772 Couplings only support Type A and C ETS Architectures with Type 1 and 2 SAE J1850 Networks.

5.8.4 The Off-Board Network Operates without the EV Network Attached

To allow the off-board systems to use the SAE J1850 Network when not connected to an EV, the electrical loading allocated to the off-board network is sufficient to allow the operation without the EV network attached.

5.8.5 The ETS Network Must Only Support One Plug-In Diagnostic Tool at a Time

Both the EV and the off-board network may allow access to the ETS network for diagnostic purposes. This presents the opportunity to connect 2 diagnostic tools to the ETS network. This would present more electrical diagnostic loading on the ETS network than SAE J1850 allows. Therefore, ETS network electrical loading allocation only supports a single test tool at a time.

However, multiple test tools are allowed if they operate in such a way as to meet the electrical loading requirements of the ETS network.

6. FUNCTIONAL SYSTEM REQUIREMENTS

6.1 Transceive Off-Board Information

This group of processes provides the means to transport information between the EV and the Off-board processes when connected. Figure 10 illustrates the relationship between these processes.

6.1.1 Process Control Requirements

None.

6.1.2 Combine EVSE Transmissions

a. Input/Output Flows

Flow Name	*Type*
EVSE_Transmission	INPUT
Off-board_Diagnostic_Transmission	INPUT
Off-board_Ext_Transmission	INPUT
Other_Off-board_Transmission	INPUT
Off-board_Transmission	OUTPUT

b. Initialization—The following shall be performed upon activation of this process:

None.

c. Operation—The following shall be performed continually while this process is active

Off-board_Transmission shall be assigned to the most dominant value of all input flows as defined in SAE J1850.

6.1.3 Transceive EVSE Information

This group of processes define the requirements to:

a. Receive messages from the SAE J1850 bus

b. Detect which messages are directed to the EVSE portion of the ETS system

c. Decode those messages to obtain information for ETS dataflows

d. Determine when to update information in other parts of the ETS System

e. Encode ETS information to send messages to the rest of the ETS system

f. Send messages to the rest of the ETS system via the SAE J1850 Bus

g. Provide basic protocols for fault recovery

h. Provide vehicle control of application communication

Figure 11 illustrates the relationship between these processes.

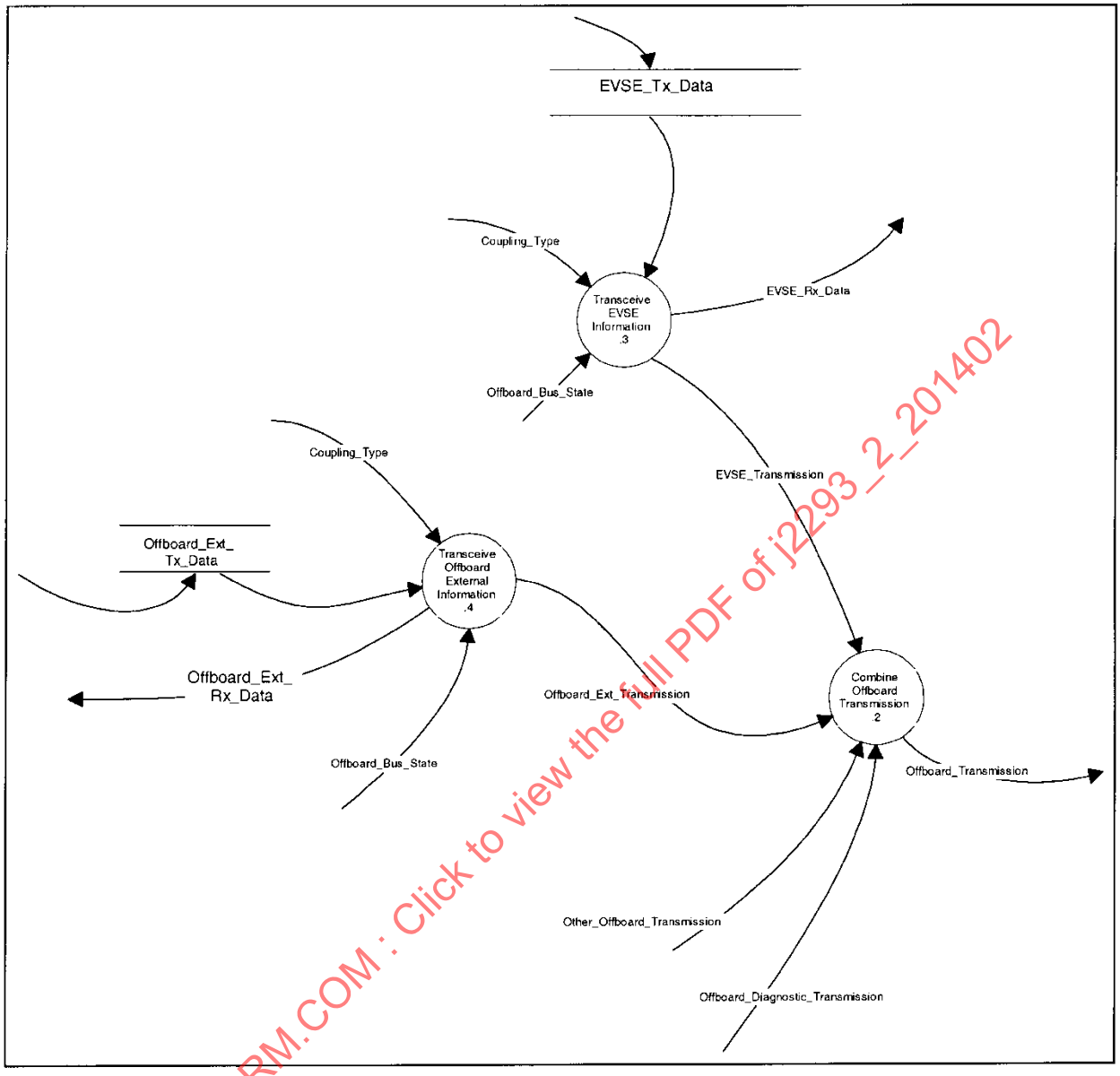

FIGURE 10 - TRANSCEIVE OFF-BOARD INFORMATION DFD

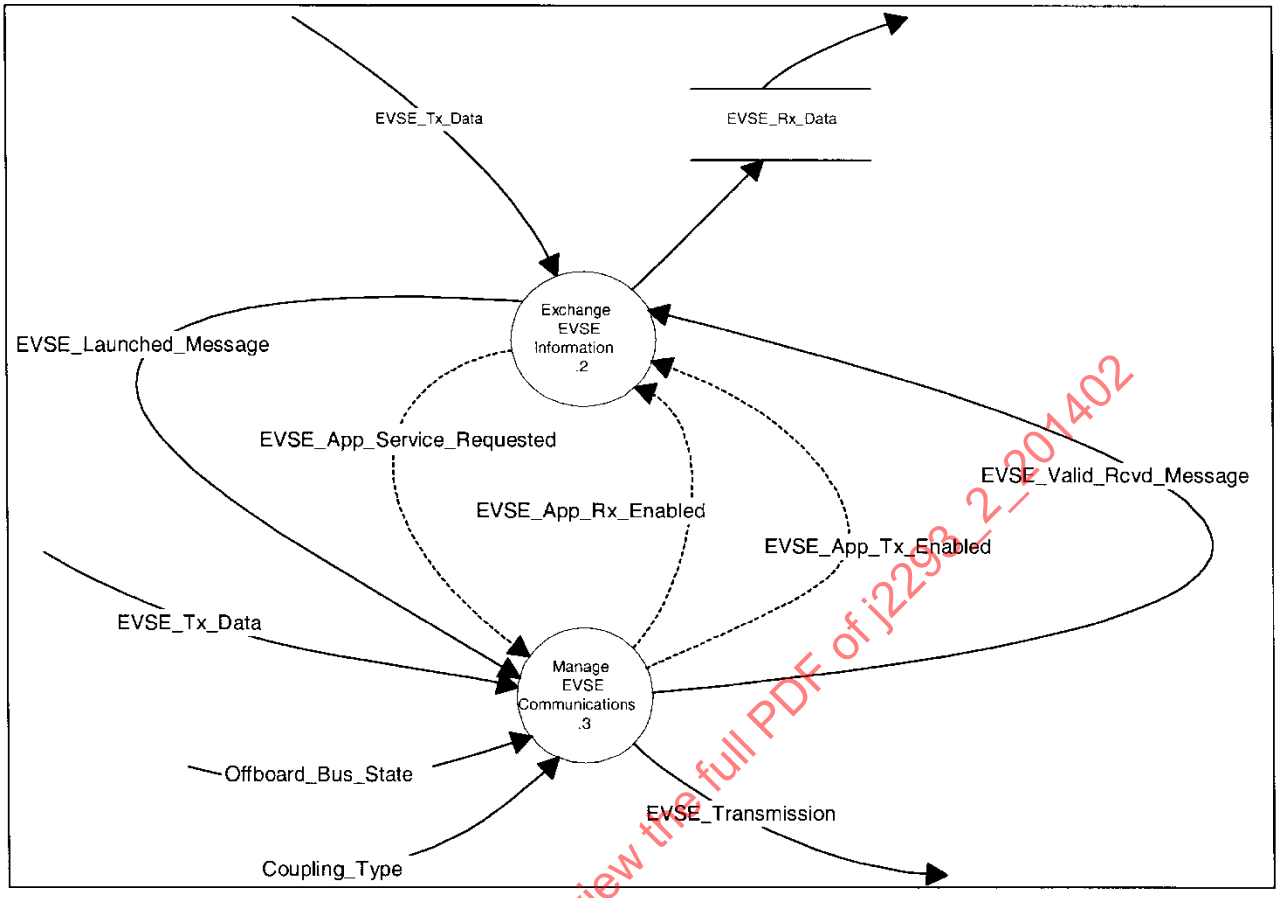

FIGURE 11 - TRANSCEIVE EVSE INFORMATION DFD

6.1.3.1 Process Control Requirements

None.

6.1.3.2 Exchange EVSE Information

These processes provide the functions to:

a. Launch the application messages

b. Receive application messages and update application data

c. Detect transmitter faults

d. Generate a request to have the application serviced

Figure 12 illustrates the relationships between the processes which perform these functions.

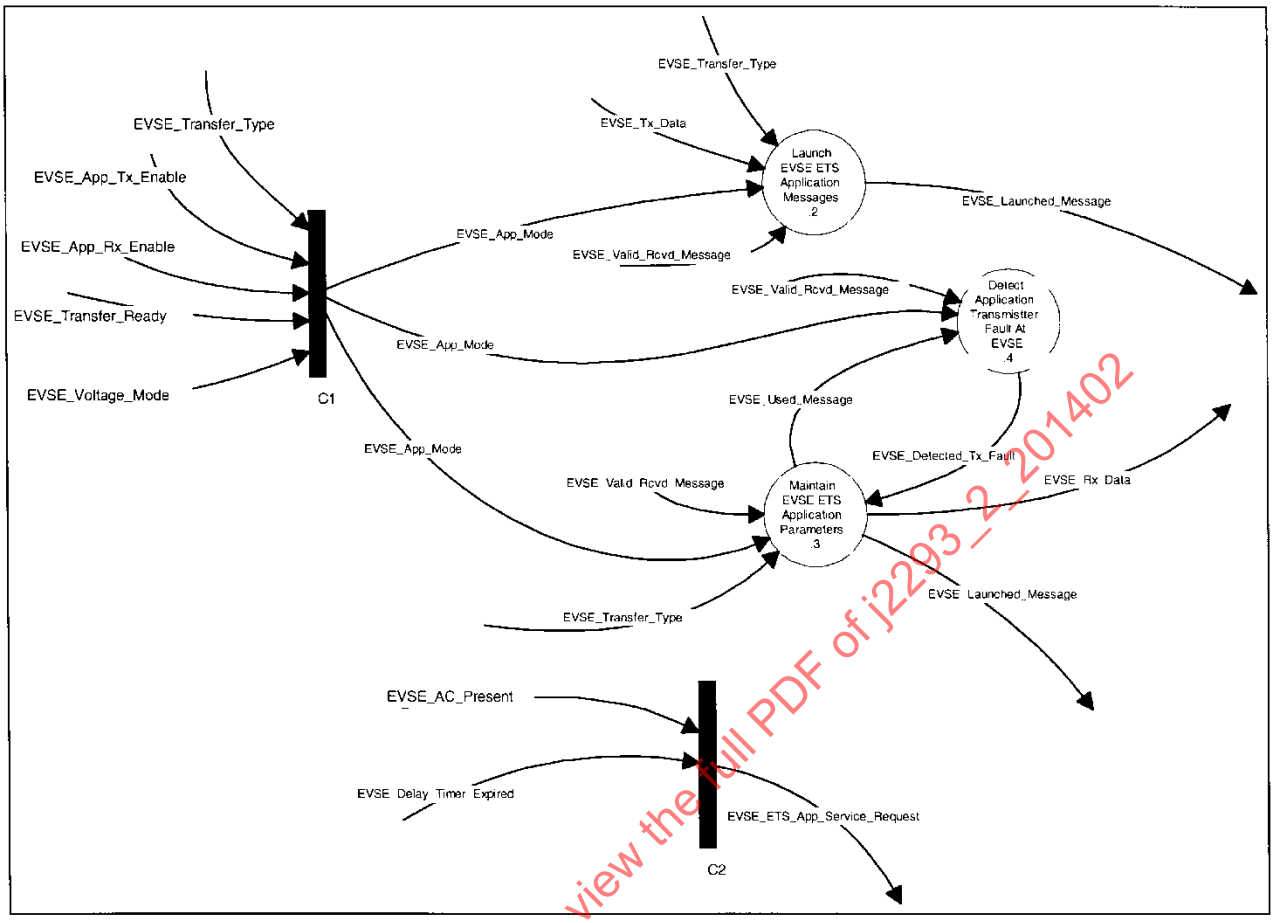

FIGURE 12 - EXCHANGE EVSE INFORMATION DFD

6.1.3.2.1 Process Control Requirements

6.1.3.2.1.1 C-spec—EVSE ETS Application Mode DT

This process uses variables from the ETS system to establish the communication mode for the ETS application. The modes of communication are defined as distinct and different when there is a significant change in message traffic sources and destination. Changes in modes are closely coupled with message transmission strategies and error recovery strategies. Table 4 provides a brief description of the ETS application communication modes.

TABLE 4 - DESCRIPTION OF APPLICATION MODES

MODE	Description
A	Comm Disabled
B	Architecture Not Selected
C	Power Control Mode
D	Voltage Control Mode

a. Input/Output Flows

Flow Name	*Type*
EVSE_App_Tx_Enabled	INPUT
EVSE_App_Rx_Enabled	INPUT
EVSE_Transfer_Type	INPUT
EVSE_Voltage_Mode	INPUT
EVSE_App_Mode	OUTPUT

b. Operation—The following shall be performed continually while the process which contains this C-spec is active:

The value of Mode shall be assigned based on the value of EVSE_App_Tx_Enabled, EVSE_App_Rx_ Enabled, EVSE_Transfer_Type, and EVSE_Voltage_Mode as defined in Table 5.

TABLE 5 - ETS NETWORK MODES DT

Input EVSE App_Tx_ Enabled	Input EVSE App_Rx_ Enabled	Input EVSE_Transfer Type = NO_TRANSFER	Input EVSE_Voltage Mode	Output EVSE_app_Mode
“Do Not Care”	FALSE	“Do Not Care”	“Do Not Care”	A
FALSE	“Do Not Care”	“Do Not Care”	“Do Not Care”	A
TRUE	TRUE	TRUE	“Do Not Care”	B
TRUE	TRUE	FALSE	FALSE	C
TRUE	TRUE	FALSE	TRUE	D

6.1.3.2.1.2 C-spec—Application Service Request STD

The EVSE must inform the EV when the delay timer expires. This must occur even if the ETS system is asleep and the network is asleep. The mechanism for any off-board application to inform the vehicle of its requirement for service is via the application service request message. This message provides a common interface for all off-board applications to inform the EV of their state. In response to this message, the EV may optionally response and enable an application. For the ETS, when the dataflow Delay_Timer_Expired transitions from FALSE to TRUE, this C-spec issues a request to the application control processes to inform the vehicle that this application requires service in the off-board network.

a. Input/Output Flows

Flow Name	*Type*
EVSE_Delay_Timer_Expired	INPUT
EVSE_AC_Present	INPUT

b. Operation—The following shall be performed continually while the process which contains this C-spec is active:

This process shall behave functionally equivalent to the State Transition Diagram shown in Figure 13.

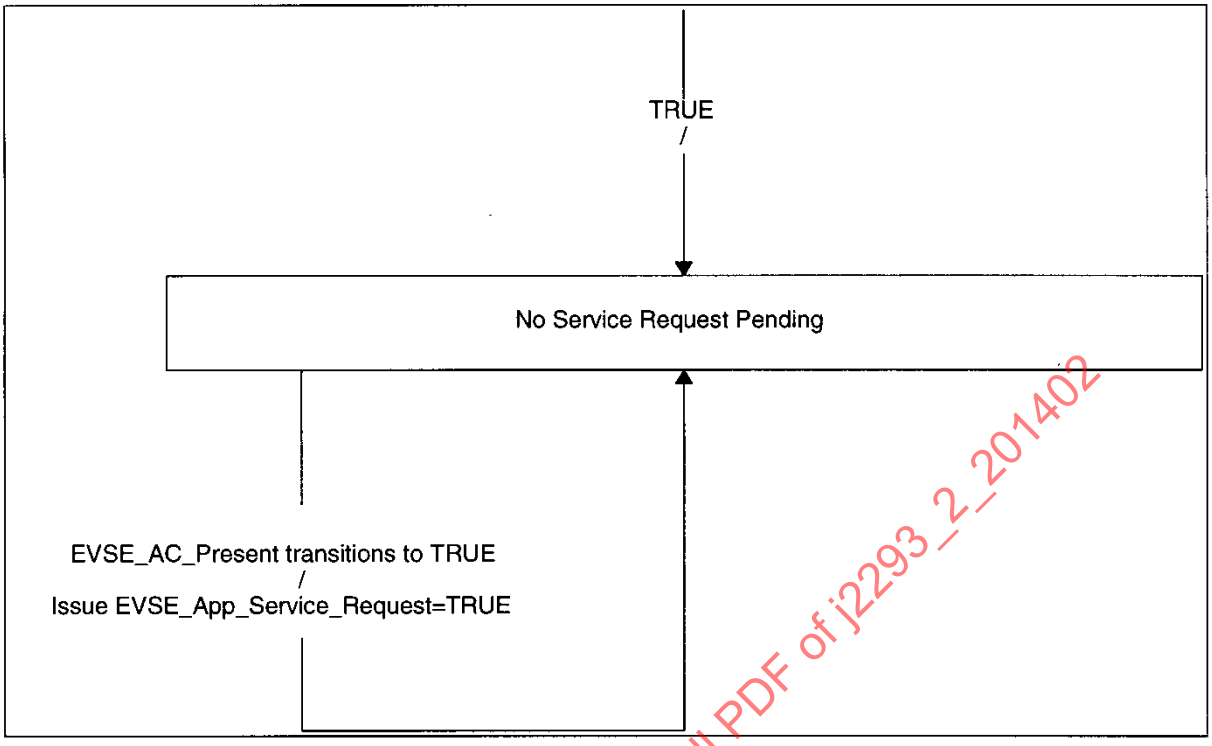

FIGURE 13 - APPLICATION SERVICE REQUEST STD

6.1.3.2.2 Launch EVSE Application Messages

This is the group of processes which launch messages based on the appropriate strategy and at the appropriate time for the ETS application.

a. *Input/Output Flows*

Flow Name	*Type*
EVSE_App_Mode	INPUT
EVSE_Valid_Rcvd_Message	INPUT
EVSE_Tx_Value	INPUT
EVSE_Launched_Message	OUTPUT
EVSE_Transfer_Type	INPUT

b. *Operation*

This process is composed of a C-spec and one process for each message sourced by the EVSE. Figure 14 provides an illustration of the general form of this process.

1. There shall be one process per message which is transmitted from the EVSE portion of the ETS. These messages are identified in Table 54A.

   Use the following procedure to determine which messages require launch processes.

   1. Select the columns in Table 54A which are under the ETS architectures supported.

   2. Every message which is marked with an 'R' under the EVSE Tx column requires a launch process.

   3. Every message which is marked with an 'O' under the EVSE Tx column may optionally have a launch process created for them based on the manufacturer's discretion.

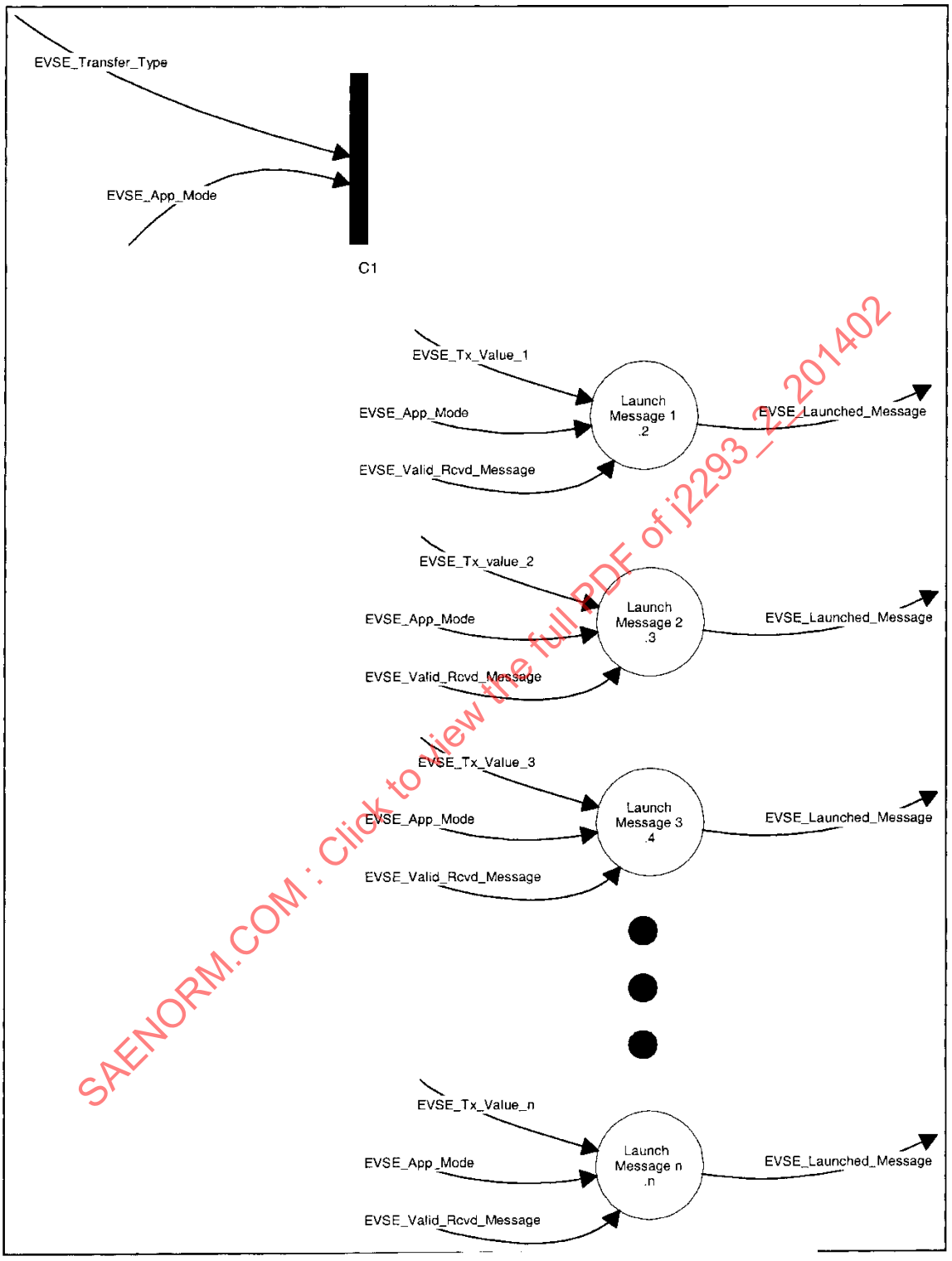

FIGURE 14 - ILLUSTRATION OF LAUNCH EVSE APPLICATION MESSAGES DFD

c. The launch process for each message shall be functionally equivalent to 6.8 with the following substitutions:

For each message identified in the previous paragraph, use the following procedure to generate the requirements for the process to launch each message.

1. For the message this processes controls, using Table 56, substitute the dataflows identified in the "Dataflows Contained" column into the context flow EVSE_Tx_Value.

2. Substitute EVSE_App_Mode for the context flow Mode.

3. Substitute EVSE_Valid_Rcvd_Message for the context flow Valid_Rcvd_Message.

4. Leave the context flow Trigger_Launch unconnected.

5. Substitute EVSE_Launched_Message for Launched_Message.

6. For the message this process controls, using Table 55 and Table 44 substitute the values for this message's launch strategy into the configuration stores in the launch parameter process template.

6.1.3.2.2.1 Process Control

a. C-spec— Launch EVSE Message PAT

1. *Input/Output Flows*

a.

Flow Name	*Type*
EVSE_App_Mode	TypeINPUT
EVSE_Transfer_Type	INPUT

b. *Processes Controlled*—All launch processes contained in 6.1.3.2.2

2. *Initialization*—The following shall be performed once upon activation of this process:

All processes controlled by the C-spec shall be disabled.

3. *Operation*—The following shall be performed continually while the process which contains this C-spec is active:

a. Every process in 6.1.3.2.2 shall be controlled via this Process Activation Table.

b. The process which corresponds to a message parameter shall be enabled for operation when the EVSE_App_Mode is equal to one of the states in Table 55 column labeled Enabled State(s) for that parameter and the message is permitted for the value of EVSE_Transfer_Type as defined in Table 54A.

6.1.3.2.3 Maintain EVSE Application Parameters

This is the group of processes which maintain a parameter's value based on the information received via network.

a. *Input/Output Flows*

Flow Name	*Type*
EVSE_App_Mode	INPUT
EVSE_Valid_Rcvd_Message	INPUT
EVSE_Detected_Tx_Fault	INPUT
EVSE_Transfer_Type	INPUT
EVSE_Used_Message	OUTPUT
EVSE_Rx_Value	OUTPUT
EVSE_Launched_Message	OUTPUT

b. *Operation*—This process is composed of a C-spec and one process for each message received by the EVSE. Figure 15 provides an illustration of the general form of this process.

1. There shall be one process per message received by the EVSE portion of the ETS. The messages in a particular system are determined by using Table 54A.

Use the following procedure to determine which messages require processes.

1. Select the columns in Table 54A which are under the ETS architectures supported.

2. Every message which is marked with an 'R' under the EVSE Rx column requires a launch process.

3. Every message which is marked with an 'O' under the EVSE Rx column may optionally have a launch process created for them at the manufacturer's discretion.

2. The process for each message shall be functionally equivalent to 6.9 with the following substitutions:

For each message identified in the previous paragraph, use the following procedure to generate the requirements for the process to launch each message.

a. For the message this process controls, using Table 56 substitute the dataflows identified in the "Dataflows Contained" column into the context flow Rx_Value.

b. Substitute EVSE_App_Mode for the context flow Mode.

c. Substitute EVSE_Valid_Rcvd_Message for the context flow Valid_Rcvd_Message.

d. Substitute EVSE_Used_Message for Used_Message.

e. Substitute EVSE_Detected_Tx_Fault for Transmitter_Fault.

f. Substitute EVSE_Launched_Message for Launched_Message.

g. For the message this process controls, using Table 55 and Table 48 substitute the values for this message's receipt strategy into the configuration stores in the maintain parameter process template.

6.1.3.2.3.1 C-spec—Maintain EVSE Application Parameters PAT

a. *Input/Output Flows*

1. *Flow Name* *Type*

EVSE_App_Mode INPUT
EVSE_Transfer_Type INPUT

2. *Processes Controlled*—All launch processes contained in 6.1.3.2.3

b. *Initialization*—The following shall be performed once upon activation of this process:

All processes controlled by this P-Spec shall be disabled.

c. *Operation*—The following shall be performed continually while the process which contains this C-spec is active:

1. Every process in 6.1.3.2.3 shall be controlled via this Process Activation Table.

2. The process which corresponds to a message parameter shall be enabled for operation when the EVSE_App_Mode is equal to one of the states in Table 55 column labeled Enabled State(s) for that parameter and the message is permitted for the value of EVSE_Transfer_Type as defined in Table 54A.

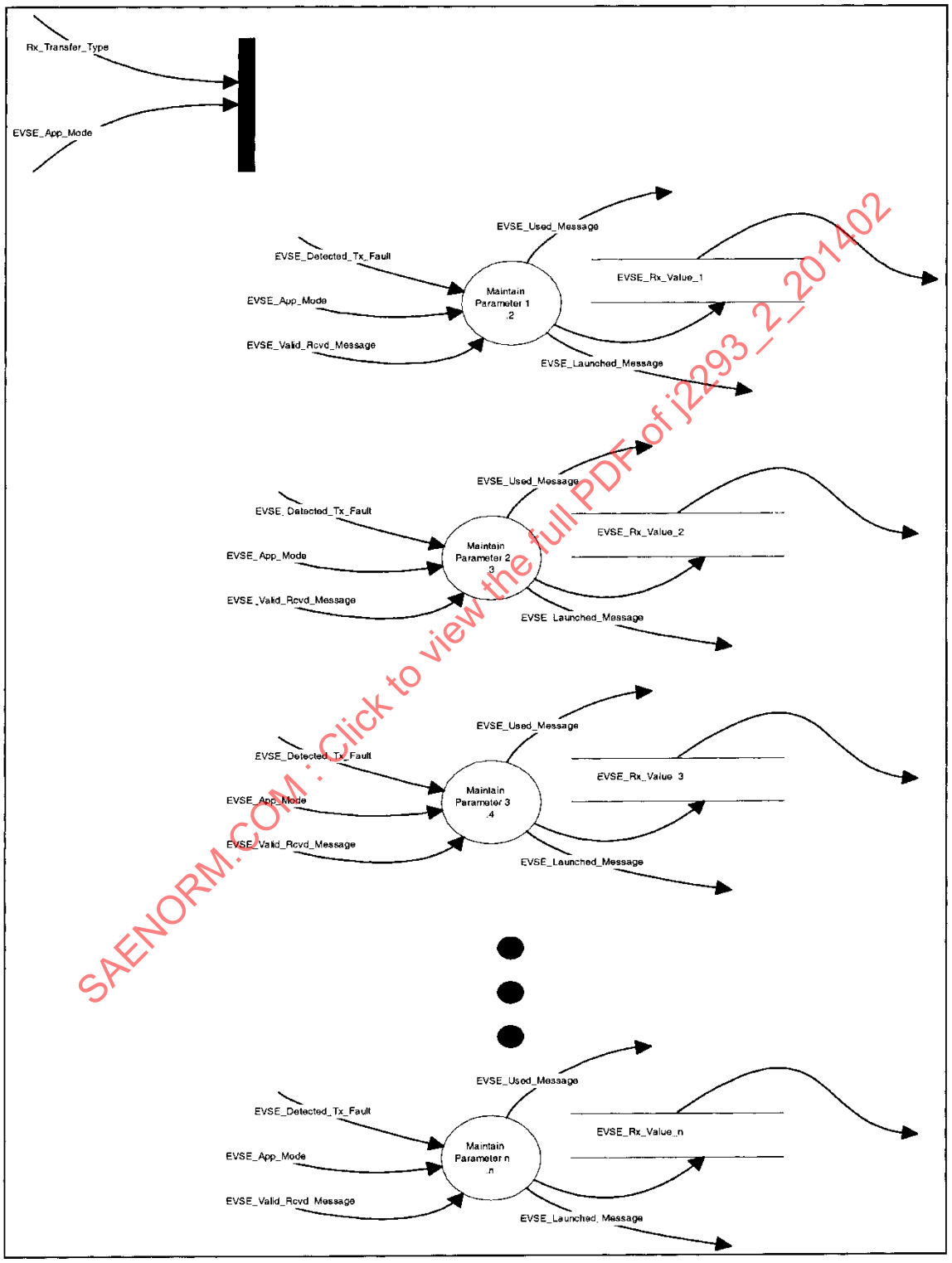

FIGURE 15 - ILLUSTRATION OF MAINTAIN EVSE APPLICATION PARAMETERS DFD

6.1.3.2.4 Detect Application Transmitter Fault

These processes learn the transmitter which are in use by the ETS system and detect faults when any of those transmitter stop transmitting. These processes reset the learning any time there is an application mode change. There are seven detectors to accommodate the following transmitters:

1. Three transmitters from the EV ETS application
2. One LMS transmitter
3. One off-board external interface transmitter
4. One EV external interface transmitter
5. One learned transmitter for future expansion

a. *Input/Output Flows*

Flow Name	*Type*
EVSE_Valid_Rcvd_Message	INPUT
EVSE_App_Mode	INPUT
EVSE_Used_Message	INPUT
EVSE_Detected_Tx_Fault	OUTPUT

b. *Operation*—This process shall perform functionally identical to 6.7 with the substitutions and exceptions which follow:

1. The context flows for 6.7 shall be substituted as defined in Table 6.

TABLE 6 - DETECT APPLICATION FAULT TEMPLATE CONTEXT FLOW SUBSTITUTIONS

Context Flow	Local Substitution Flow
Used_Message	EVSE_Used_Message
Detected_Tx_Fault	EVSE_Detected_Tx_Fault
Mode	EVSE_App_Mode
Valid_Rcvd_Message	EVSE_Valid_Rcvd_Message

6.1.3.3 Manage EVSE Communications

a. *Input/Output Flows*

Flow Name	*Type*
EVSE_App_Service_Request	INPUT
EVSE_App_Tx_Enabled	OUTPUT
EVSE_App_Rx_Enabled	OUTPUT
EVSE_Launched_Messsage	INPUT
EVSE_Tx_Data	INPUT
Off-board_Bus_State	INPUT
EVSE_Valid_Rcvd_Message	OUTPUT
Coupling_Type	INPUT
EVSE_Transmission_State	OUTPUT

b. *Operation*—This process shall perform functionally identical to 6.5 with the substitutions and exceptions which follow:

1. The context flows for 6.5 shall be substituted as defined in Table 7.

TABLE 7 - MANAGE EVSE COMMUNICATIONS CONTEXT FLOW SUBSTITUTIONS

Context Flow	Local Substitution Flow
App_Rx_Enabled	EVSE_App_Rx_Enabled
App_Service_Request	EVSE_App_Service_Request
App_Tx_Enabled	EVSE_App_Tx_Enabled
Bus_State	Off-board_Bus_State
Coupling_Type	Coupling_Type
Launched_Message	EVSE_Launched_Message
Location	OFF-BOARD
Transmission	EVSE_Transmission
Tx_Data	EVSE_Tx_Data
Valid_Primary_IDs	hexadecimal 70, hexadecimal 71, hexadecimal FE, hexadecimal FF
Valid_Rcvd_Message	EVSE_Valid_Rcvd_Message

6.1.4 Transceive Off-Board External ETS Information

This group of processes define the requirements to:

a. Receive messages from the SAE J1850 bus

b. Detect which messages are directed to the external portion of the ETS which is off-board

c. Decode those messages to obtain information for ETS dataflows

d. Determine when to update information in other parts of the ETS System

e. Encode ETS information to send messages to the rest of the ETS system

f. Send messages to the rest of the ETS system via the SAE J1850 Bus

g. Provide basic protocols for fault recovery

h. Provide vehicle control of application communication

Figure 16 illustrates the relationship between these processes.

6.1.4.1 Process Control Requirements

None.

6.1.4.2 Exchange Off-Board Ext Application Information

These processes provide the functions to:

a. Launch the application messages

b. Receive application messages and update application data

c. Detect transmitter faults

d. Generate a request to have the application serviced

Figure 17 illustrates the relationships between the processes which perform these functions.

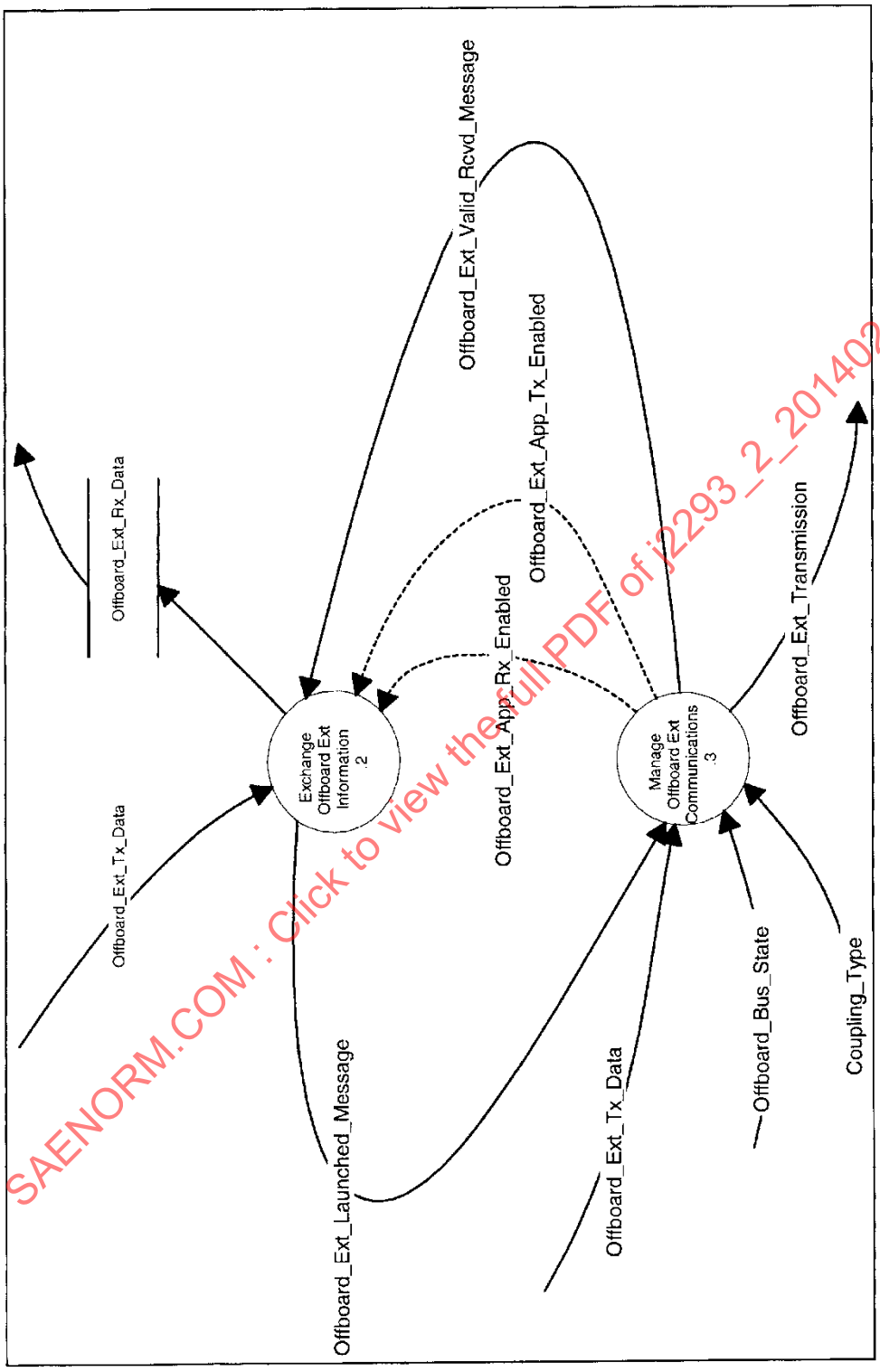

FIGURE 16 - TRANSCEIVE OFF-BOARD EXT INFORMATION DFD

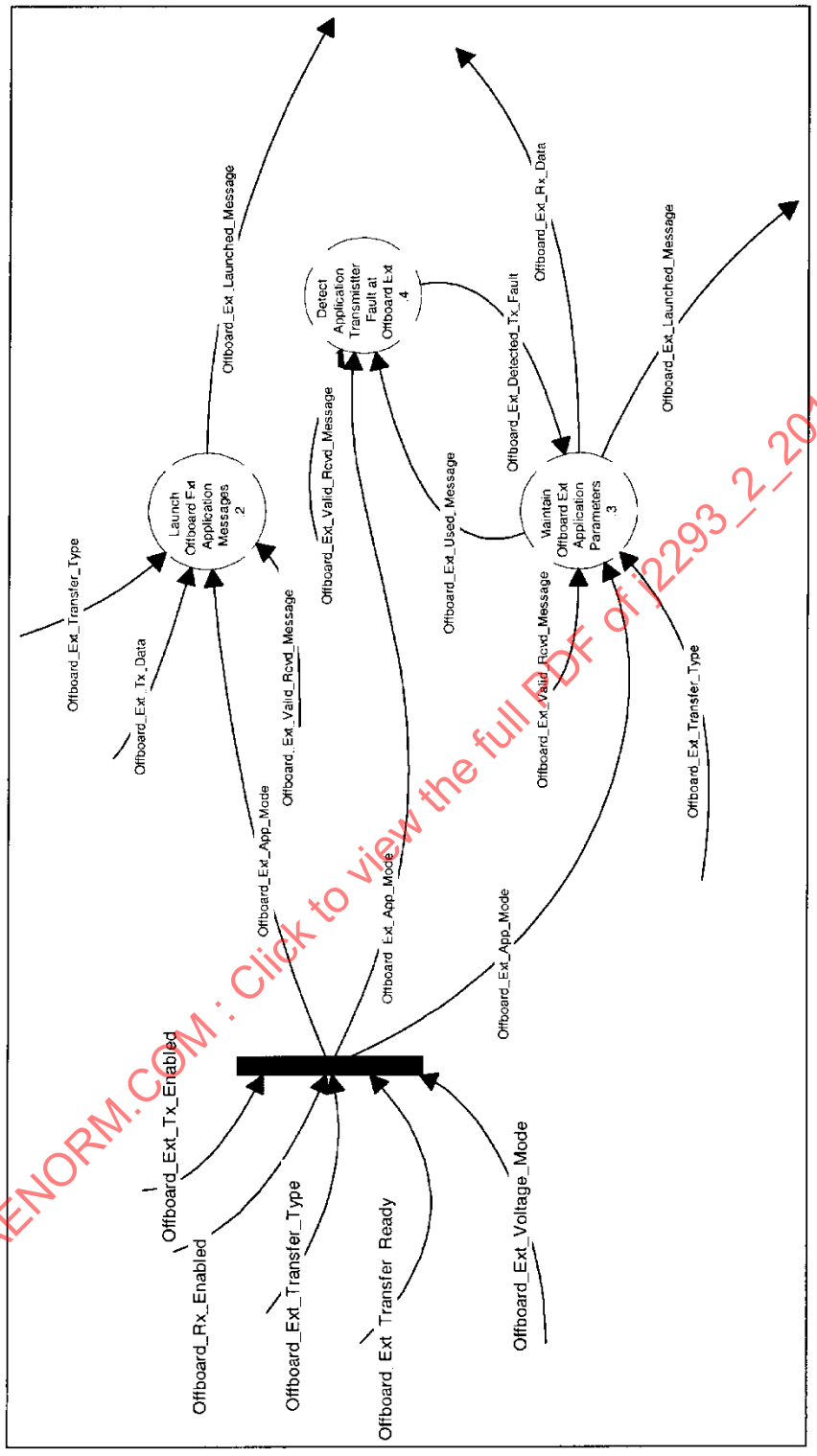

FIGURE 17 - EXCHANGE OFF-BOARD EXT INFORMATION

6.1.4.2.1 Process Control Requirements

6.1.4.2.1.1 C-spec—Off-board Ext Application Mode DT

This process uses variables from the ETS system to establish the communication mode for the ETS application. The modes of communication are defined as distinct and different when there is a significant change in message traffic sources and destination. Changes in modes are closely coupled with message transmission strategies and error recovery strategies. Table 4 provides a brief description of the ETS application communication modes.

a. *Input/Output Flows*

Flow Name	*Type*
Off-board_Ext_App_Tx_Enabled	INPUT
Off-board_Ext_App_Rx_Enabled	INPUT
Off-board_Ext_Transfer_Type	INPUT
Off-board_Ext_Voltage_Mode	INPUT
Off-board_Ext_App_Mode	OUTPUT

b. *Operation*—The following shall be performed continually while the process which contains this C-spec is active:

The value of Mode shall be assigned based on the value of Off-board_Ext_App_Tx_Enabled, Off-board_Ext_App_Rx_Enabled, Off-board_Ext_Transfer_Type and Off-board_Ext_Voltage_Mode as defined in Table 8.

TABLE 8 - OFF-BOARD_EXT NETWORK MODES DT

Input Off-board_Ext_ App_Tx_ Enabled	Input Off-board_Ext_ App_Rx_ Enabled	Off-board_Ext_ Transfer_Type+ NO_TRANSFER	Input Off-board_Ext Voltage Mode	Output Off-board_Ext_ MODE
Do Not Care	FALSE	**Do Not Care**	**Do Not Care**	A
FALSE	**Do Not Care**	**Do Not Care**	**Do Not Care**	A
TRUE	TRUE	TRUE	**Do Not Care**	B
TRUE	TRUE	FALSE	FALSE	C
TRUE	TRUE	FALSE	TRUE	D

6.1.4.2.2 Launch Off-Board Ext Application Messages

This is the group of processes which launch messages based on the appropriate strategy and at the appropriate time for the ETS application.

a. *Input/Output Flows*

Flow Name	*Type*
Off-board_Ext_App_Mode	INPUT
Off-board_Ext_Valid_Rcvd_Message	INPUT
Off-board_Ext_Tx_Value	INPUT
Off-board_Ext_Launched_Message	OUTPUT
Off-board_Ext_Transfer_Type	INPUT

b. *Operation*—This process is composed of a C-spec and one process for each message sourced by the EVSE. Figure 18 provides an illustration of the general form of this process.

1. There shall be one process per message which is transmitted from the EVSE portion of the ETS. These messages are identified in Table 54A.

   Use the following procedure to determine which messages require launch processes to be created for them.

   a. Select the columns in Table 54A which are under the Off-board_Ext.

   b. Every message which is marked with an 'R' under the Off-board_Ext Tx column requires a launch process.

   c. Every message which is marked with an 'O' under the Off-board_Ext Tx column may optionally have a launch process created for them based on the manufacturer's discretion.

2. The launch process for each message shall be functionally equivalent to 6.8 with the following substitutions.

   For each message identified in the previous paragraph, use the following procedure to generate the requirements for the process to launch each message.

   a. For the message this process controls, using Table 56 substitute the dataflows identified in the “Dataflows Contained” column into the context flow Tx_Value.

   b. Substitute Off-board_Ext _App_Mode for the context flow Mode.

   c. Substitute Off-board_Ext _Valid_Rcvd_Message for the context flow Valid_Rcvd_Message.

   d. Leave the context flow Trigger_Launch unconnected.

   e. Substitute Off-board_Ext _Launched_Message for Launched_Message.

   f. For the message this process controls, using Table 55 and Table 54 substitute the values for this message's launch strategy into the configuration stores in the launch parameter process template.

6.1.4.2.2.1 Process Control

a. C-spec—Launch Off-board Ext Message PAT

   1. *Input/Output Flows*

      a.

Flow Name	*Type*
Off-board_Ext_App_Mode	INPUT
Off-board_Ext_Transfer_Type	INPUT

      b. *Processes Controlled*—All launch processes contained in 6.1.3.2.2.

   2. *Initialization*—The following shall be performed once upon activation of this process:

      All processes controlled by the C-spec shall be disabled.

   3. *Operation*—The following shall be performed continually while the process which contains this C-spec is active:

      a. Every process in 6.1.3.2.2 shall be controlled via this Process Activation Table.

      b. The process which corresponds to a message parameter shall be enabled for operation when the Off-board_Ext_App_Mode is equal to one of the states in Table 55 column labeled Enabled State(s) for that parameter and the message is permitted for the value of Off-board_Ext_Transfer_Type as defined in Table 54A.

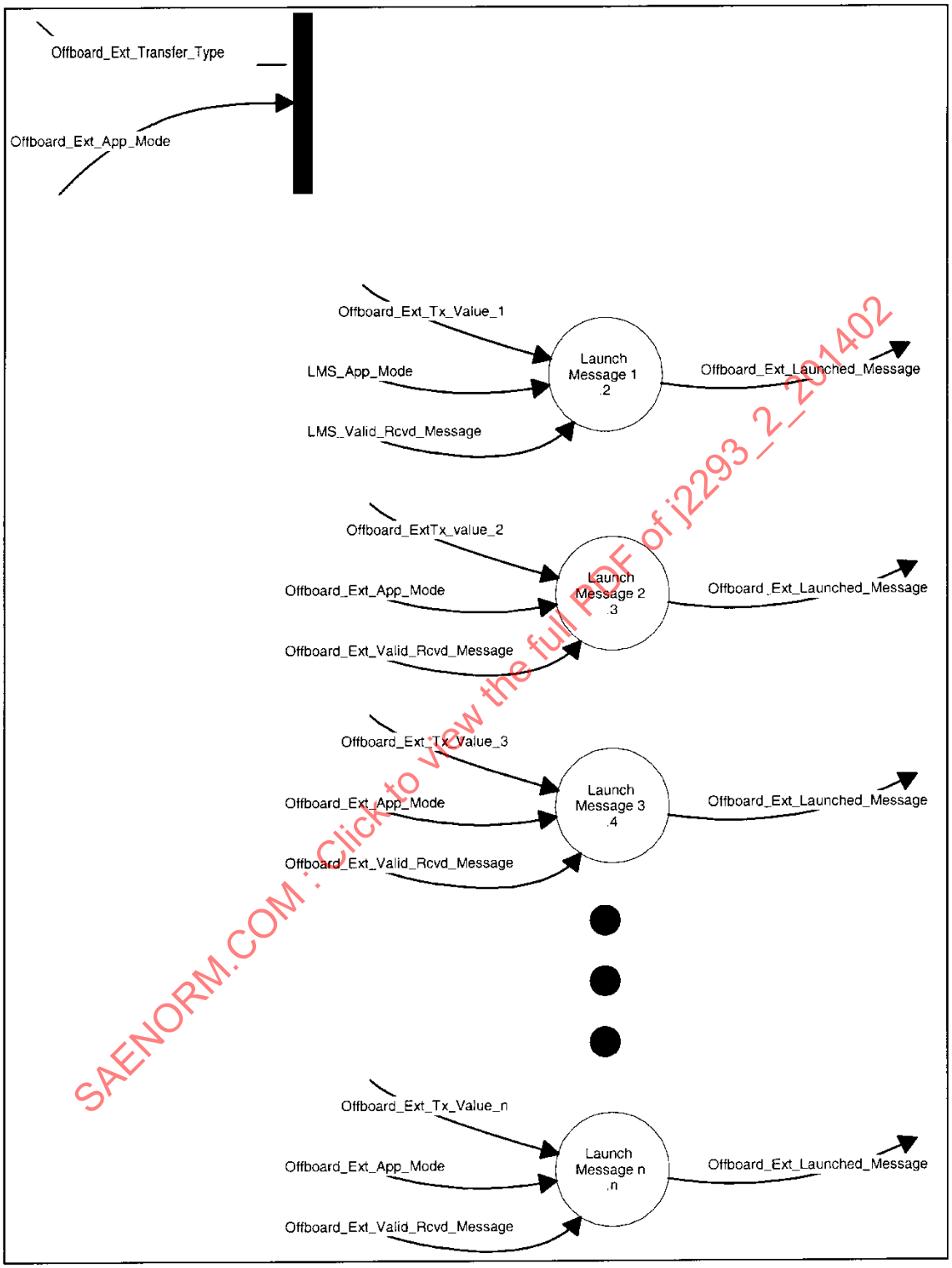

FIGURE 18 - ILLUSTRATION OF LAUNCH OFF-BOARD EXT APPLICATION MESSAGES DFD

6.1.4.2.3 Maintain Application Parameters

This is the group of processes which maintain a parameter's value based on the information received via network.

a. *Input/Output Flows*

Flow Name	*Type*
Off-board_Ext_App_Mode	INPUT
Off-board_Ext_Valid_Rcvd_Message	INPUT
Off-board_Ext_Detected_Tx_Fault	INPUT
Off-board_Ext_Transfer_Type	INPUT
Off-board_Ext_Used_Message	OUTPUT
Off-board_Ext_Rx_Value	OUTPUT
Off-board_Ext_Launched_Message	OUTPUT

b. *Operation*—This process is composed of a C-spec and one process for each message received by the Off-board External systems. Figure 19 provides an illustration of the general form of this process.

1. There shall be one process per message received by the Off-board External portion of the ETS. The messages in a particular system are determined by using Table 54A.

   Use the following procedure to determine which messages require processes.

   a. Select the columns in Table 54A which are under the Off-board_Ext.

   b. Every message which is marked with an 'R' under the Off-board_Ext Rx column requires a maintain process.

   c. Every message which is marked with an 'O' under the Off-board_Ext Rx column may optionally have a maintain process created at the manufacturer's discretion.

2. The process for each message shall be functionally equivalent to 6.9 with the following substitutions:

   For each message identified in the previous paragraph, use the following procedure to generate the requirements for the process to maintain the parameters in each message.

   a. For the message this process controls, using Table 56 substitute the dataflows identified in the "Dataflows Contained" column into the context flow Rx_Value.

   b. Substitute Off-board_Ext _Valid_Rcvd_Message for the context flow Valid_Rcvd_Message.

   c. Substitute Off-board_Ext _Used_Message for Used_Message.

   d. Substitute Off-board_Ext _Detected_Tx_Fault for Transmitter_Fault.

   e. Substitute Off-board_Ext _Launched_Message for Launched_Message.

   f. For the message this process controls, using Table 55 and Table 48 substitute the values for this message's receipt strategy into the configuration stores in the maintain parameter process template.

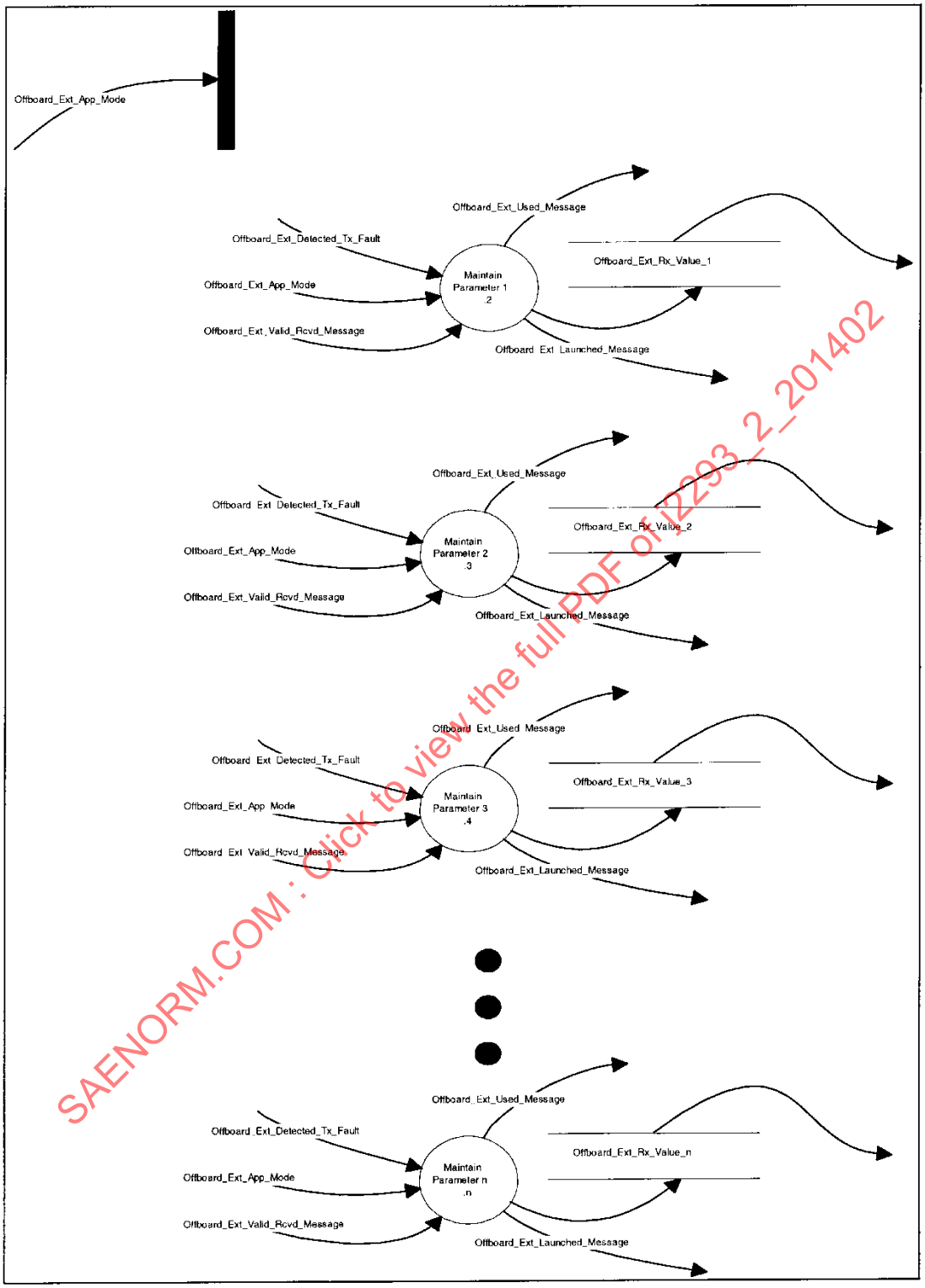

FIGURE 19 - ILLUSTRATION OF MAINTAIN OFF-BOARD EXT APPLICATION PARAMETERS DFD

6.1.4.2.3.1 C-spec—Maintain Off-board Ext Application Parameters PAT

a. Input/Output Flows

1. Flow Name *Type*
Off-board_Ext_App_Mode INPUT

2. Process Controlled—All launch processes contained in 6.1.3.2.3

b. *Initialization*—The following shall be performed once upon activation of this process.

All processes controlled by this P-spec shall be disabled.

c. Operation—The following shall be performed once upon activation of this process:

a. Every process in 6.1.3.2.3 shall be controlled via this Process Activation Table.

b. The process which corresponds to a message parameter shall be enabled for operation when the Off-board_Ext_App_Mode is equal to one of the states in Table 55 column labeled Enabled State(s) for that parameter and the message is permitted for the value of Off-board_Ext_Transfer_Type as defined in Table 54A.

6.1.4.2.4 Detect Application Transmitter Fault at Off-Board Ext

These processes learn the transmitter which are in use by the ETS system and detect faults when any of those transmitters stop transmitting. These processes reset the learning any time there is an application mode change. There are seven detectors to accommodate the following transmitters:

1. Three transmitters from the EV ETS application
2. One LMS transmitter
3. One off-board external interface transmitter
4. One EV external interface transmitter
5. One learned transmitter for future expansion

a. *Input/Output Flows*

Flow Name	*Type*
Off-board_Ext_Valid_Rcvd_Message	INPUT
Off-board_Ext_App_Mode	INPUT
Off-board_Ext_Used_Message	INPUT
Off-board_Ext_Detected_Tx_Fault	OUTPUT

b. *Operation*—This process shall perform functionally identical to 6.7 with the substitutions and exceptions which follow.

1. The context flows for 6.7 shall be substituted as defined in Table 6.

TABLE 9 - DETECT APPLICATION FAULT TEMPLATE CONTEXY FLOW SUBSTITUTIONS

Context Flow	Local Substitution Flow
Used_Message	Off-board_Ext_Used_Message
Detected_Tx_Fault	Off-board_Ext_Detected_Tx_Fault
Mode	Off-board_Ext_App_Mode
Valid_Rcvd_Message	Off-board_Ext_Valid_Rcvd_Message

6.1.4.3 Manage Off-Board Ext Communications

a. *Input/Output Flows*

Flow Name	*Type*
Off-board_Ext_App_Service_Request	INPUT
Off-board_Ext_App_Tx_Enabled	OUTPUT
Off-board_Ext_App_Rx_Enabled	OUTPUT
Off-board_Ext_Launched_Message	INPUT
Off-board_Ext_Tx_Data	INPUT
Off-board_Bus_State	INPUT
Off-board_Ext_Valid_Rcvd_Message	OUTPUT
Coupling_Type	INPUT
Off-board_Ext_Transmission_State	OUTPUT

b. *Operation*—This process shall perform functionally identical to 6.5 with the substitutions and exceptions which follow.

1. The context flows for 6.5 shall be substituted as defined in Table 10.

TABLE 10 - MANAGE OFF-BOARD EXT COMMUNICATIONS CONTEXT FLOW SUBSTITUTIONS

Context Flow	Local Substitution Flow
App_Rx_Enabled	Off-board_Ext_App_Rx_Enabled
App_Service_Request	Off-board_Ext_App_Service_Request
App_Tx_Enabled	Off-board_Ext_App_Tx_Enabled
Bus_State	Off-board_Bus_State
Coupling_Type	Coupling_Type
Launched_Message	Off-board_Ext_Launched_Message
Location	Off-board
Transmission	Off-board_Ext_Transmission
Tx_Data	Off-board_Ext_Tx_Data
Valid_Primary_IDs	hexadecimal 70, hexadecimal 71, hexadecimal FE, hexadecimal FF
Valid_Rcvd_Message	Off-board_Ext_Valid_Rcvd_Message

6.2 Connect ETS Networks

These processes provide the functions to connect the EV and the Off-board networks to form the ETS Network. Figure 20 illustrates the relationship between the processes which provide these functions.

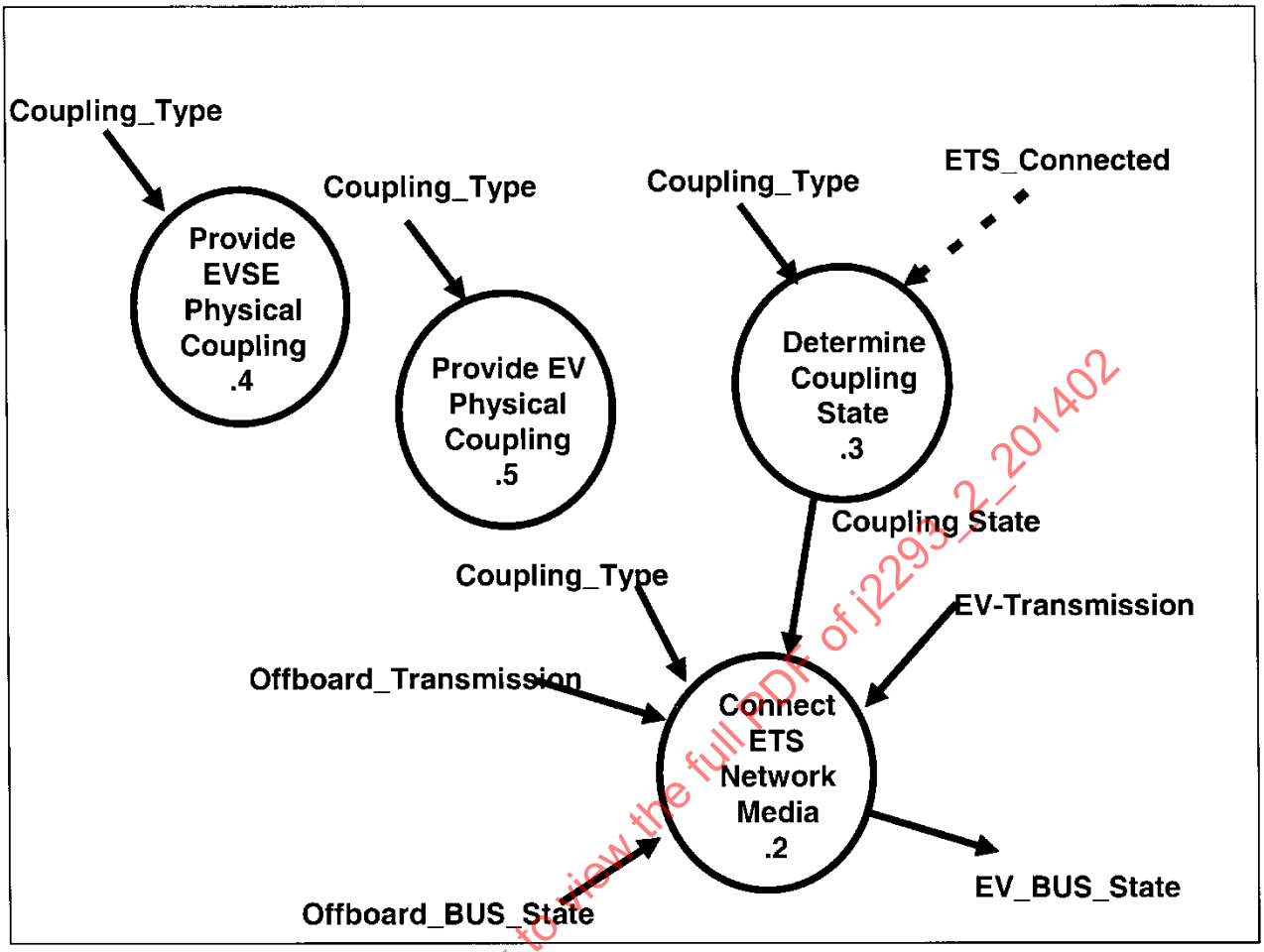

FIGURE 20 - CONNECT ETS NETWORKS DFD

6.2.1 Process Control Requirements

None.

6.2.2 Connect ETS Network Media

These processes provide the functions of combining the network transmission based on the state of the coupling. Figure 21 illustrates the relationship between these processes.

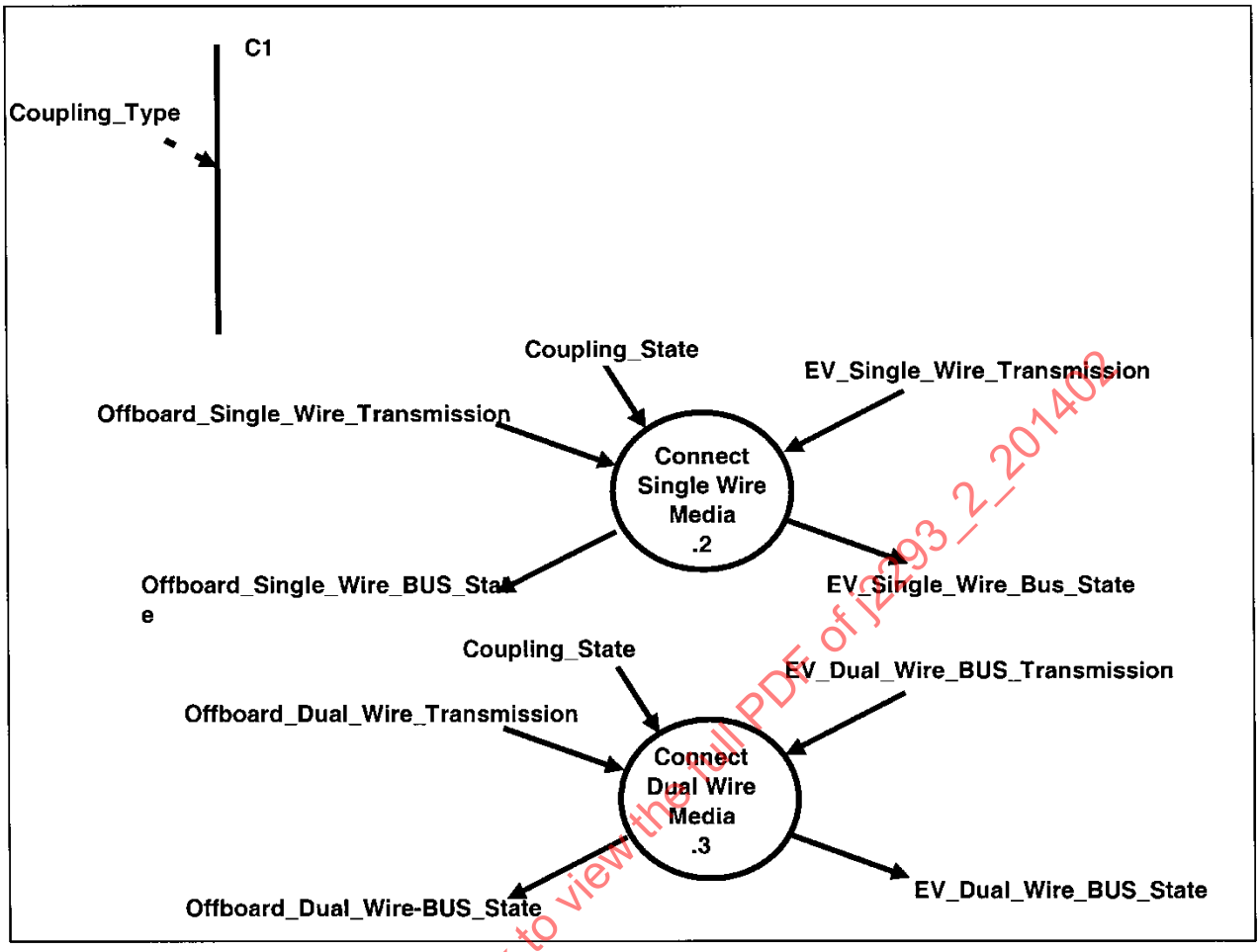

FIGURE 21 - CONNECT ETS NETWORK MEDIA DFD

6.2.2.1 Coupling Process Control Requirements

a. *Input/Output Flows*

1. *Flow Name* — *Type*
   Coupling_Type — INPUT

2. *Process Controlled*—6.2.2.2

b. *Initialization*—The following shall be performed once upon activation of this process:

All processes controlled by this C-spec shall be disabled.

c. *Operation*—The following shall be performed continually while the process which contains this C-spec is active:

The process identified in Table 11 shall be enabled and disabled based on the state of the control flow in this table.

TABLE 11 - PAT 2.1P1

Control Flow Coupling_Type	Process 6.2.2.2 Connect Dual Wire Media
SAE J1773	Disabled
SAE J1772	Enabled

6.2.2.1.1 Connect Single Wire Media

a. *Input/Output Flows*

Flow Name	*Type*
Coupling State	INPUT
EV_Single_Wire_Bus_State	INPUT
Off-board_Single_Wire_Bus_State	INPUT
EV_Single_Wire_Transmission	OUTPUT
Off-board_Single_Wire_Transmission	OUTPUT

b. *Initialization*—None.

c. *Operation*—The following shall be performed continually while this process is active. See Table 12.

TABLE 12 - SINGLE WIRE STATE ARBITRATION TABLE

Input Coupling State	Input Off-board_Single_ Wire_ Transmission	Input EV_Single_Wire_ Transmission	Output Off-board_Single_ Wire_Bus_State	EV_Single_Wire_ Bus_State
Not_Connected	Unbiased or Passive	Unbiased or Passive	Passive	Passive
		Active	Passive	Active
	Active	Unbiased or Passive	Active	Passive
		Active	Active	Active
SAE J1772_Connected or SAE J1773_Connected	Unbiased or Passive	Unbiased or Passive	Passive	Passive
		Active	Active	Active
	Active	Unbiased or Passive	Active	Active
		Active	Active	Active

6.2.2.2 Connect Dual Wire Media

a. *Input/Output*

Flow Name	*Type*
Coupling State	INPUT
EV_Dual_Wire_Bus_State	INPUT
Off-board_Dual_Wire_Bus_State	INPUT
EV_Dual_Wire_Transmission	OUTPUT
Off-board_Dual_Wire_Transmission	OUTPUT

b. *Operation*—The following shall be performed continually while this process is active. See Table 13:

TABLE 13 - DUAL WIRE STATE ARBITRATION TABLE

Input Coupling State	Input Off-board_Dual_ Wire_ Transmission	Input EV_Dual_Wire_ Transmission	Output Off-board_Dual_ Wire_Bus_State	EV_Dual_Wire_ Bus_State
Not_Connected or	Unbiased or Passive	UnBiased or Passive Active	Passive Passive	Passive Active
SAE J1773_Connected	Active	Unbiased or Passive Active	Active Active	Passive Active
SAE J1772_Connected	Unbiased or Passive	Unbiased or Passive Active	Passive Active	Passive Active
	Active	Unbiased or Passive Active	Active Active	Active Active
	Unbiased or Passive	Unbiased or Passive	Passive	Passive

6.2.3 Determine Coupling State

a. *Input/Output*

Flow Name	*Type*
Coupling_Type	INPUT
ETS_Connected	INPUT
Coupling_State	OUTPUT

b. *Initialization*—None.

c. *Operation*—The following shall be performed continually while this process is active:

Coupling_State shall have a value based on Coupling_Type and ETS_Connected as shown in Table 14.

TABLE 14 - COUPLING STATE TABLE

Input Coupling_Type	Input ETS_Connected	Output Coupling_State
Don't Care	FALSE	Not_Connected
SAE J1772	TRUE	J1772_Connected
SAE J1773	TRUE	J1773_Connected

6.2.4 Provide EVSE Physical Coupling

6.2.4.1 Coupling Type

The EVSE shall provide for coupling to either a SAE J1772 or SAE J1773 compatible coupler.

6.2.4.2 SAE J1772 Coupling

The EVSE shall use both the Type 1 and Type 2 physical media for communication. Where the EV and EVSE use a SAE J1772 coupling, the range of values for the vehicle and off-board network electrical values changes. This occurs because the SAE J1850 network is required to be fully functional in both components when they are separated and connected. This requires that some portion of the electrical loading allocation of the vehicle be allocated to the EV and some portion allocated to the EVSE. The paragraphs which follows define this allocation of the standard SAE J1850 values.

6.2.4.2.1 Type 1 Media (Single Wire Media)

When connected to an EV, the EVSE shall provide a SAE J1850 electrically compatible implementation with the exceptions documented in Table 15:

TABLE 15 - SAE J1772 - TYPE 1 MEDIA - EVSE NETWORK REQUIREMENTS

Parameter Name	Units	Min	Typical	Max
Off-board Network Rload	Ohms	1325	n/a	1575
Off-board Network Cload	Picofarads	2470	n/a	4500
Off-board Wire Length	meters	n/a	n/a	15
Off-board Service Tool Wire Length	meters	n/a	n/a	5
Maximum # of EVSE Unit Loads		1	n/a	8
Off Vehicle Network Length	meters	n/a	n/a	n/a
Off Vehicle Resistance	Ohms	n/a	n/a	n/a
Off Vehicle Capacitance	Picofarads	n/a	n/a	n/a

6.2.4.2.2 Type 2 Media (Dual Wire Media)

When connected to an EV the EVSE shall provide a SAE J1850 electrically compatible implementation with the following exceptions in Table 16:

TABLE 16 - SAE J1772 - TYPE 1 MEDIA - EVSE NETWORK REQUIREMENTS

Parameter Name	Units	Min	Typical	Max
Network Rload	Ohms	200	n/a	337
EVSE Wire Length	meters	n/a	n/a	15
EVSE Service Tool Wire Length	meters	n/a	n/a	5
Number EVSE Nodes		1	n/a	8
Number EVSE Service Nodes		1	1	1

6.2.4.2.3 SAE J1773 Coupling

When connected to an EV, the EVSE shall provide a SAE J1850 compatible implementation with the exceptions documented in Table 17. Where the EV and EVSE use a SAE J1773 coupling, the range of values for the vehicle and off-board network electrical values change. Most of the electrical values for the network are not applicable because since the EVSE and EV are galvanically isolated. However, the introduction of an RF transceiver in the communication path creates some new requirements. The paragraphs which follows define these changes to standard SAE J1850 values.

TABLE 17 - J1773 EVSE NETWORK REQUIREMENTS

Parameter Name	Units	Min	Typical	Max
Transceiver Latency	microseconds	0	n/a	4
Transceiver Blanking	microseconds	0	n/a	4
SAE J1850 Node Noise Rejection - Minimum Pulse Filtering	microseconds	8	n/a	n/a
Off Vehicle Network Length	meters	n/a	n/a	n/a
Total Vehicle Network Length	meters	n/a	n/a	n/a
Maximum Number of Standard Unit Loads	—	n/a	n/a	n/a
Off Vehicle Resistance	ohms	n/a	n/a	n/a
Off Vehicle Capacitance	ohms	n/a	n/a	n/a
Input High Voltage	volts	n/a	n/a	n/a
Input Low Voltage	volts	n/a	n/a	n/a
Output High Voltage	volts	n/a	n/a	n/a
Output Low Voltage	volts	n/a	n/a	n/a
Absolute Ground Offset Voltage	volts	n/a	n/a	n/a
Network Resistance	ohms	n/a	n/a	n/a
Network Capacitance	picofarads	n/a	n/a	n/a
Network Time Constant	microseconds	n/a	n/a	n/a
Signal Transition Time	microseconds	n/a	n/a	n/a
Node Resistance	ohms	n/a	n/a	n/a
Node Capacitance	picofarads	n/a	n/a	n/a
Node Leakage Current	micro amps	n/a	n/a	n/a

6.2.5 Provide EV Physical Coupling

6.2.5.1 Coupling Type

The EV shall provide for coupling to either a SAE J1772 or SAE J1773 compatible coupler.

6.2.5.2 SAE J1772 Coupling

The EV shall utilize either the Type 1 or Type 2 physical media for communication.

6.2.5.2.1 Type 1 Media (Single Wire Media)

When connected to an EVSE, the EV shall provide a SAE J1850 electrically compatible implementation with the exceptions in Table 18.

TABLE 18 - SAE J1772 - TYPE 1 MEDIA EV NETWORK REQUIREMENTS

Parameter Name	Units	Min	Typical	Max
EV Network Rload	Ohms	433	n/a	1575
EV Network Cload	Picofarads	2470	n/a	12000
EV Wire Length	meters	n/a	n/a	20
EV Service Tool Wire Length	meters	n/a	n/a	5
Number of EV Unit Loads		1	n/a	24
Off Vehicle Network Length	meters	n/a	n/a	20
Off Vehicle Resistance	Ohms	1175	n/a	n/a
Off Vehicle Capacitance	picofarads	n/a	n/a	5000

6.2.5.2.2 Type 2 Media (Dual Wire Media)

When connected to an EVSE, the EV shall provide a SAE J1850 electrically compatible implementation with the exceptions in Table 19:

TABLE 19 - SAE J1772 - TYPE 2 MEDIA, EV NETWORK REQUIREMENTS

Parameter Name	Units	Min	Typical	Max
Network Rload	Ohms	146	n/a	337
EV Wire Length	meters	n/a	n/a	15
EV Service Tool Wire Length	meters	n/a	n/a	5
Number EV Nodes		1	n/a	22
Number EV Service Nodes		1	1	1

6.2.5.3 SAE J1773 Coupling

When connected to an EVSE, the EV shall provide a SAE J1850 compatible implementation with the exceptions documented in Table 20.

TABLE 20 - SAE J1773 EV NETWORK REQUIREMENTS

Parameter Name	Units	Min	Typical	Max
Transceiver Latency	microseconds	0	n/a	41
Transceiver Blanking	microseconds	0	n/a	41
SAE J1850 Node False Edge Rejection - Minimum Edge Filtering	microseconds	9	n/a	n/a
Off Vehicle Network Length	meters	n/a	n/a	n/a
Total Vehicle Network Length	meters	n/a	n/a	n/a
Maximum Number of Standard Unit Loads	-	n/a	n/a	n/a
Off Vehicle Resistance	ohms	n/a	n/a	n/a
Off Vehicle Capacitance	ohms	n/a	n/a	n/a
Input High Voltage	volts	n/a	n/a	n/a
Input Low Voltage	volts	n/a	n/a	n/a
Output High Voltage	volts	n/a	n/a	n/a
Output Low Voltage	volts	n/a	n/a	n/a
Absolute Ground Offset Voltage	volts	n/a	n/a	n/a
Network Resistance	ohms	n/a	n/a	n/a
Network Capacitance	picofarads	n/a	n/a	n/a
Network Time Constant	microseconds	n/a	n/a	n/a
Signal Transition Time	microseconds	n/a	n/a	n/a
Node Resistance	ohms	n/a	n/a	n/a
Node Capacitance	picofarads	n/a	n/a	n/a
Node Leakage Current	micro amps	n/a	n/a	n/a

6.3 Transceive EV Information

This group of processes provides the means to transport information between the EV and the Off-board processes when connected.

Figure 22 provides an illustration of the relationship between these processes.

6.3.1 Process Control Requirements

None.

6.3.2 Combine EV Transmissions

a. *Input/Output Flows*

Flow Name	*Type*
EV_ETS_Transmission	INPUT
EV_Ext_Transmission	INPUT
Other_EV_Transmission	INPUT
EV_Diagnostic_Transmission	INPUT
EV_Transmission	OUTPUT

b. *Initialization*—The following shall be performed upon activation of this process:

None.

c. *Operation*—The following shall be performed continually while this process is active:

EV_Transmission shall be assigned to the most dominant value of all input flows as defined in SAE J1850.

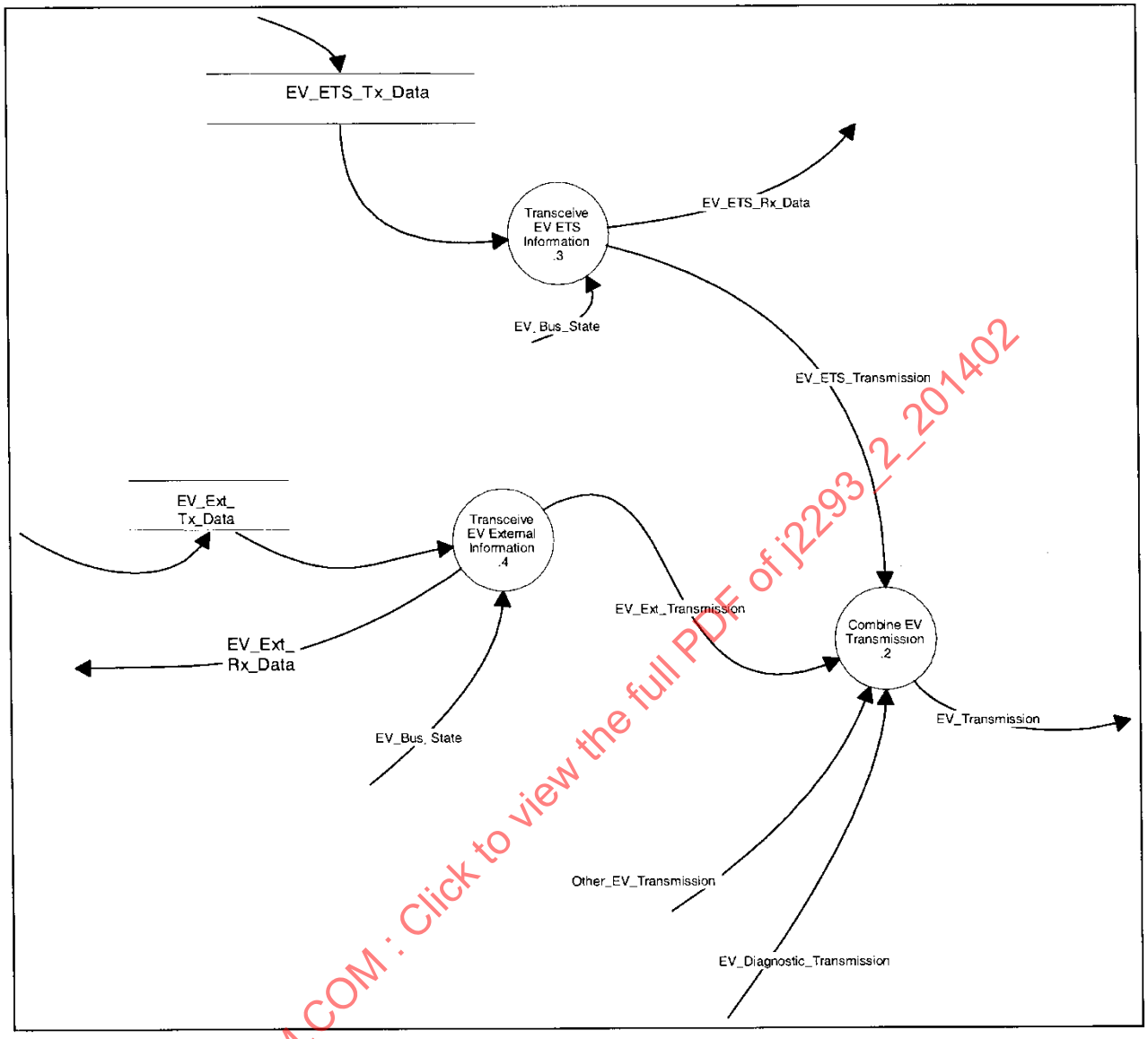

FIGURE 22 - TRANSCEIVE EV INFORMATION DFD

6.3.3 Transceive EV ETS Information

This group of processes define the requirements to:

a. Receive messages from the SAE J1850 bus

b. Detect which messages are directed to the EV portion of the ETS system

c. Decode those messages to obtain information for ETS dataflows

d. Determine when to update information in other parts of the ETS System

e. Encode ETS information to send messages to the rest of the ETS system

f. Send messages to the rest of the ETS system via the SAE J1850 Bus

g. Provide basic protocols for fault recovery

h. Provide vehicle control of application communication

Figure 23 illustrates the relationship between these processes.

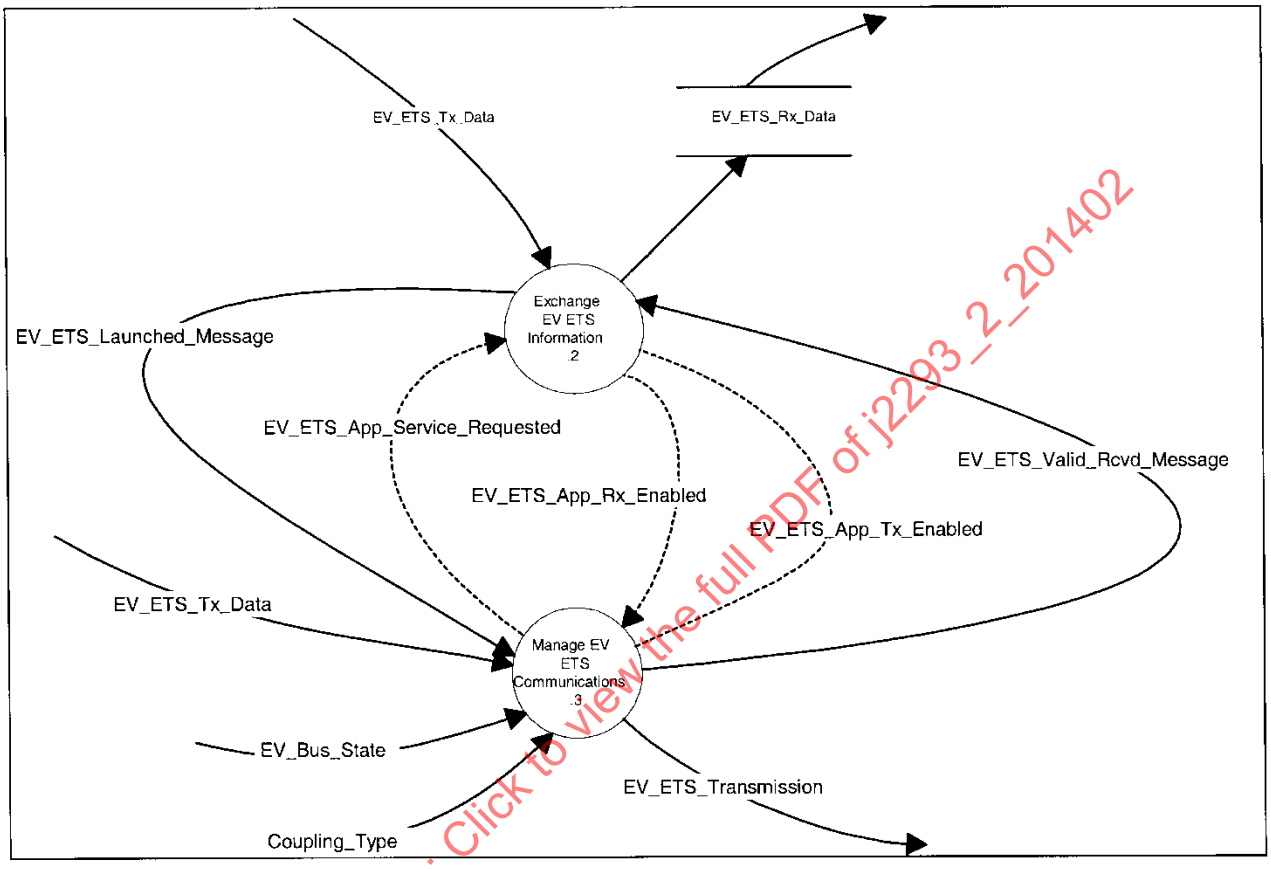

FIGURE 23 - TRANSCEIVE EV ETS INFORMATION DFD

6.3.3.1 Process Control Requirements

None.

6.3.3.2 Exchange EV ETS Information

These processes provide the functions to:

a. Launch the application messages

b. Receive application messages and update application data

c. Detect transmitter faults

d. Generate a request to have the application serviced

Figure 24 illustrates the relationships between the processes which perform these functions.

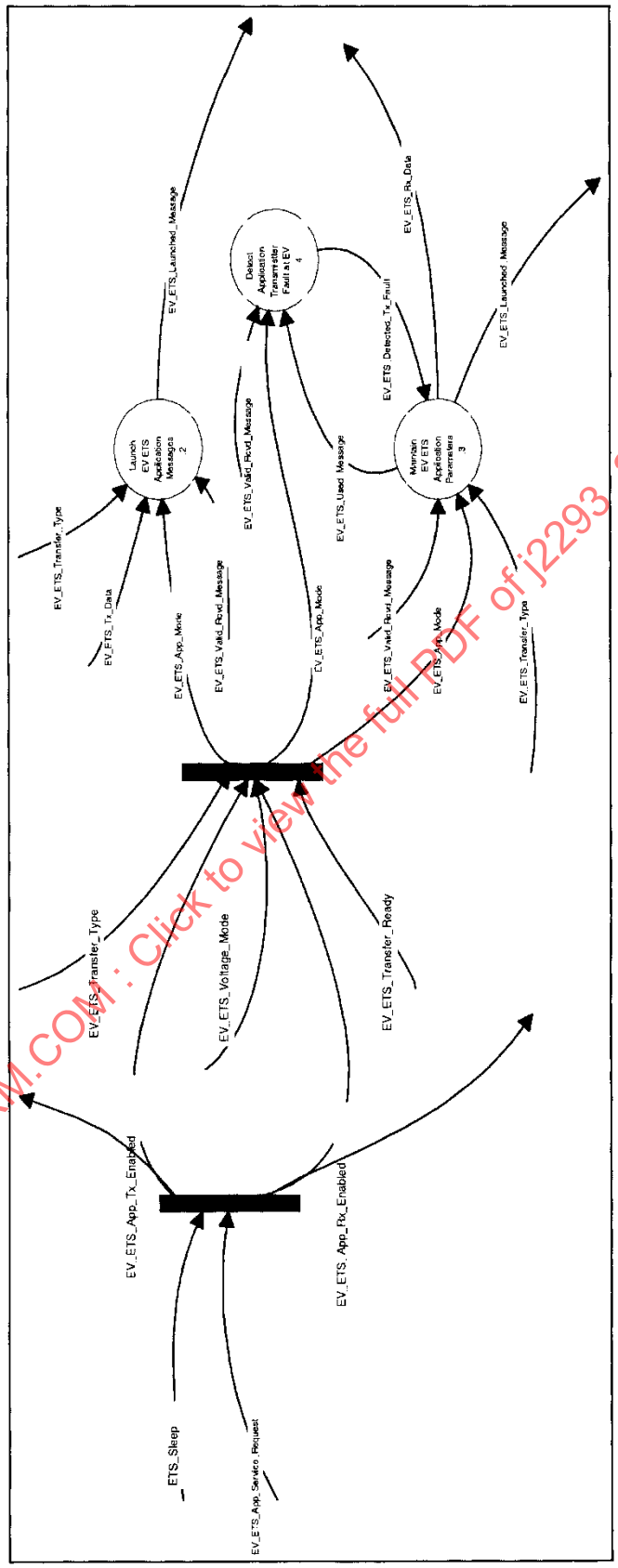

FIGURE 24 - EXCHANGE EV ETS INFORMATION DFD

6.3.3.2.1 Process Control Requirements

6.3.3.2.1.1 C-spec—EV ETS Application Mode DT

This process uses variables from the ETS system to establish the communication mode for the ETS application. The modes of communication are defined as distinct and different when there is a significant change in message traffic sources and destination. Changes in modes are closely coupled with message transmission strategies and error recovery strategies. Table 4 provides a brief description of the ETS application communication modes.

a. *Input/Output Flows*

Flow Name	*Type*
EV_ETS_App_Tx_Enabled	INPUT
EV_ETS_App_Rx_Enabled	INPUT
EV_ETS_Transfer_Type	INPUT
EV_ETS_Voltage_Mode	INPUT
EV_ETS_App_Mode	OUTPUT

b. *Operation*—The following shall be performed continually while the process which contains this C-spec is active:

The value of Mode shall be assigned based on the value of EV_ETS_App_Tx_Enabled, EV_ETS_ App_Rx_Enabled, EV_ETS_Transfer_Type and EV_ETS_Voltage_Mode as defined in Table 21.

TABLE 21 - EV ETS NETWORK MODES DT

Input EV_ETS_ App_Tx_ Enabled	Input EV_ETS_ App_Rx_ Enabled	Input EV_ETS_ Transfer Type = NO_TRANSFER	Input EV_ETS Voltage Mode	Output EV_ETS MODE
Do Not Care	FALSE	**Do Not Care**	**Do Not Care**	A
FALSE	**Do Not Care**	**Do Not Care**	**Do Not Care**	A
TRUE	TRUE	TRUE	**Do Not Care**	B
TRUE	TRUE	FALSE	FALSE	D
TRUE	TRUE	FALSE	TRUE	E

6.3.3.2.1.2 C-spec—EV ETS Comm State STD

The EVSE must inform the EV when the delay timer expires. This must occur even if the ETS system is asleep and the network is asleep. The mechanism for any off-board application to inform the vehicle of its requirement for service is via the application service request message. This message provides a common interface for all off-board applications to inform the EV of their state. In response to this message, the EV may optionally response and enable an application. For the ETS, when the dataflow Delay_Timer_Expired transitions from FALSE to TRUE, this C-spec issues a request to the application control processes to inform the vehicle that this application requires service in the off-board network

a. *Input/Output Flows*

Flow Name	*Type*
EV_ETS_Sleep	INPUT
EV_ETS_App_Service_Request	INPUT
EV_ETS_App_Tx_Enable	OUTPUT
EV_ETS_App_Rx_Enable	OUTPUT

b. *Operation*—The following shall be performed continually while the process which contains this C-spec is active:

This process shall behave functionally equivalent to the State Transition Diagram shown in Figure 25.

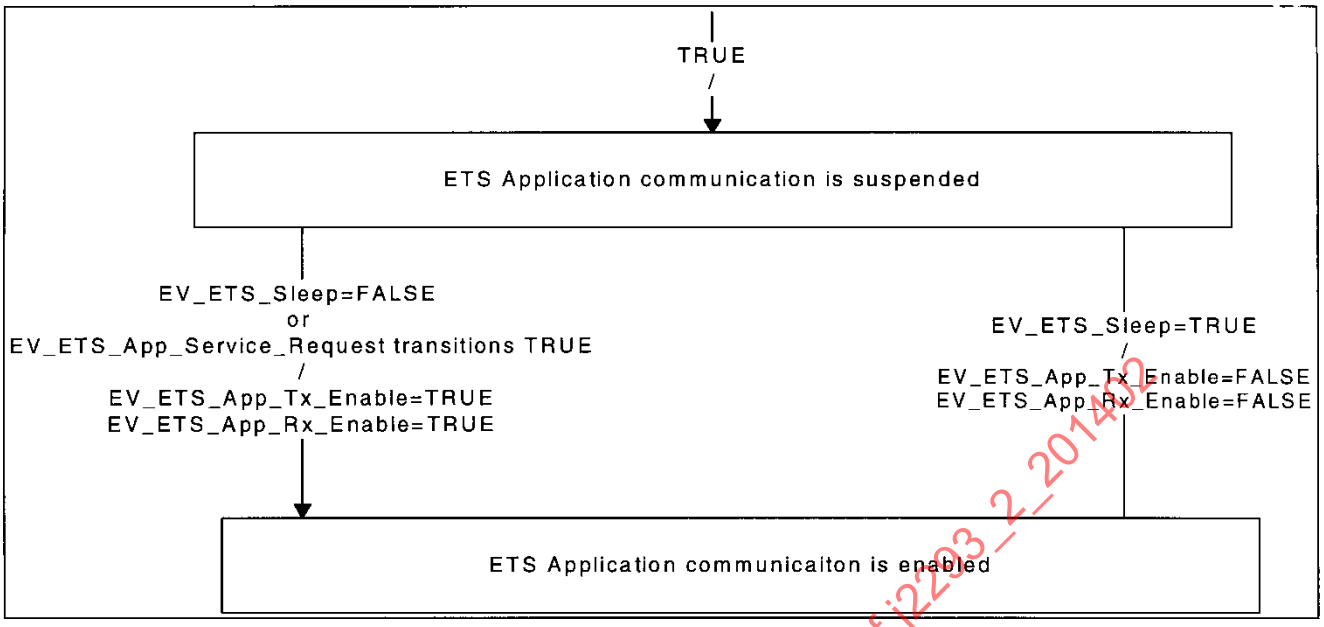

FIGURE 25 - EV ETS COMM STATE STD

6.3.3.2.2 Launch EV ETS Application Messages

This is the group of processes which launch messages based on the appropriate strategy and at the appropriate time for the ETS application.

a. *Input/Output Flows*

Flow Name	*Type*
EV_ETS_App_Mode	INPUT
EV_ETS_Valid_Rcvd_Message	INPUT
EV_ETS_Tx_Value	INPUT
EV_ETS_Launched_Message	OUTPUT
EV_ETS_Transfer_Type	INPUT

b. *Operation*—This process is composed of a C-spec and one process for each message sourced by the EV ETS. Figure 26 provides an illustration of the general form of this process.

1. There shall be one process per message which is transmitted from the EVSE portion of the ETS. These messages are identified in Table 54A.

   Use the following procedure to determine which messages require launch processes to be created for them.

   a. Select the columns in Table 54A which are under the ETS architectures supported.

   b. Every message which is marked with an 'R' under the EV Tx column requires a launch process.

   c. Every message which is marked with an 'O' under the EV Tx column may optionally have a launch process created for them based on the manufacturer's discretion.

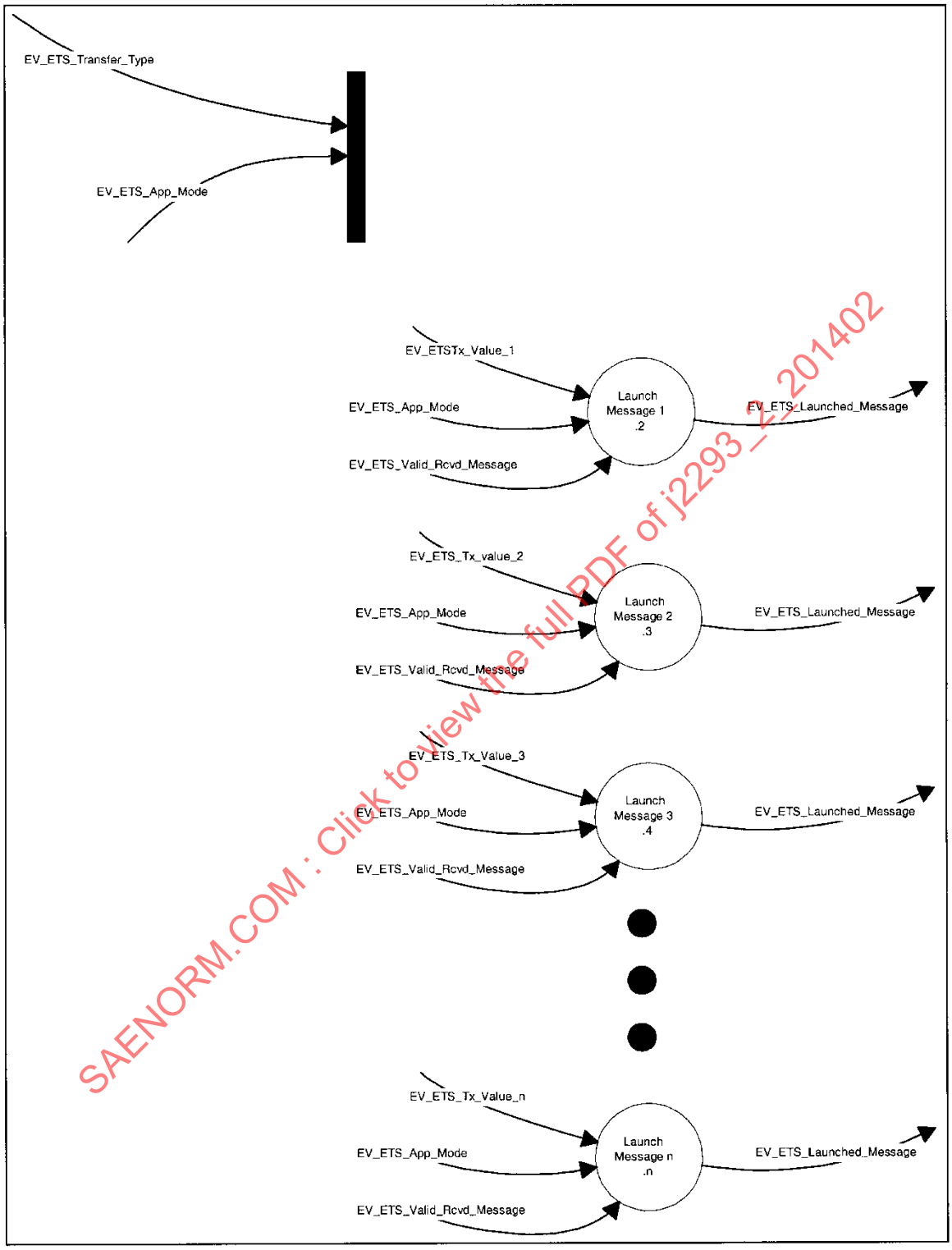

FIGURE 26 - ILLUSTRATION OF LAUNCH EV ETS APPLICATION MESSAGES DFD

2. The launch process for each message shall be functionally equivalent to 6.8 with the following substitutions:

   For each message identified in the previous paragraph, use the following procedure to generate the requirements for the process to launch each message.

   a. For the message this processes controls, using Table 56 substitute the dataflows identified in the "Dataflows Contained" column into the context flow Tx_Value.

   b. Substitute EV_ETS_App_Mode for the context flow Mode.

   c. Substitute EV_ETS_Valid_Rcvd_Message for the context flow Valid_Rcvd_Message.

   d. Leave the context flow Trigger_Launch unconnected.

   e. Substitute EV_ETS_Launched_Message for Launched_Message.

   f. For the message this process controls, using Table 55 and Table 44 substitute the values for this message's launch strategy into the configuration stores in the launch parameter process template.

6.3.3.2.2.1 Process Control Requirements

1. **C-spec—EV ETS Launch Message PAT**

   a. *Input/Output Flows*

      1. *Flow Name* — *Type*
         EV_ETS_App_Mode — INPUT

      2. *Processes Controlled*—All launch processes contained in 6.1.3.2.2

   b. *Initialization*—The following shall be performed once upon activation of this process:

      All processes controlled by the C-spec shall be disabled.

   c. *Operation*—The following shall be performed continually while to process which contains this C-spec is active:

      1. Every process in 6.1.3.2.2 shall be controlled via this Process Activation Table.

      2. The process which corresponds to a message parameter shall be enabled for operation when the EVSE_App_Mode is equal to one of the states in Table 55 column labeled Enabled State(s) for that parameter and the message is permitted for the value of EV_ETS_Transfer_Type as defined in Table 54A.

6.3.3.2.3 Maintain Application Parameters

This is the group of processes which maintain a parameter's value based on the information received via network.

a. *Input/Output Flows*

Flow Name	*Type*
EV_ETS_App_Mode	INPUT
EV_ETS_Valid_Rcvd_Message	INPUT
EV_ETS_Detected_Tx_Fault	INPUT
EV_ETS_Transfer_Type	INPUT
EV_ETS_Used_Message	OUTPUT
EV_ETS_Rx_Value	OUTPUT
EV_ETS_Launched_Message	OUTPUT

b. *Operation*—This process is composed of a C-spec and one process for each message received by the EVSE. Figure 27 provides an illustration of the general form of this process.

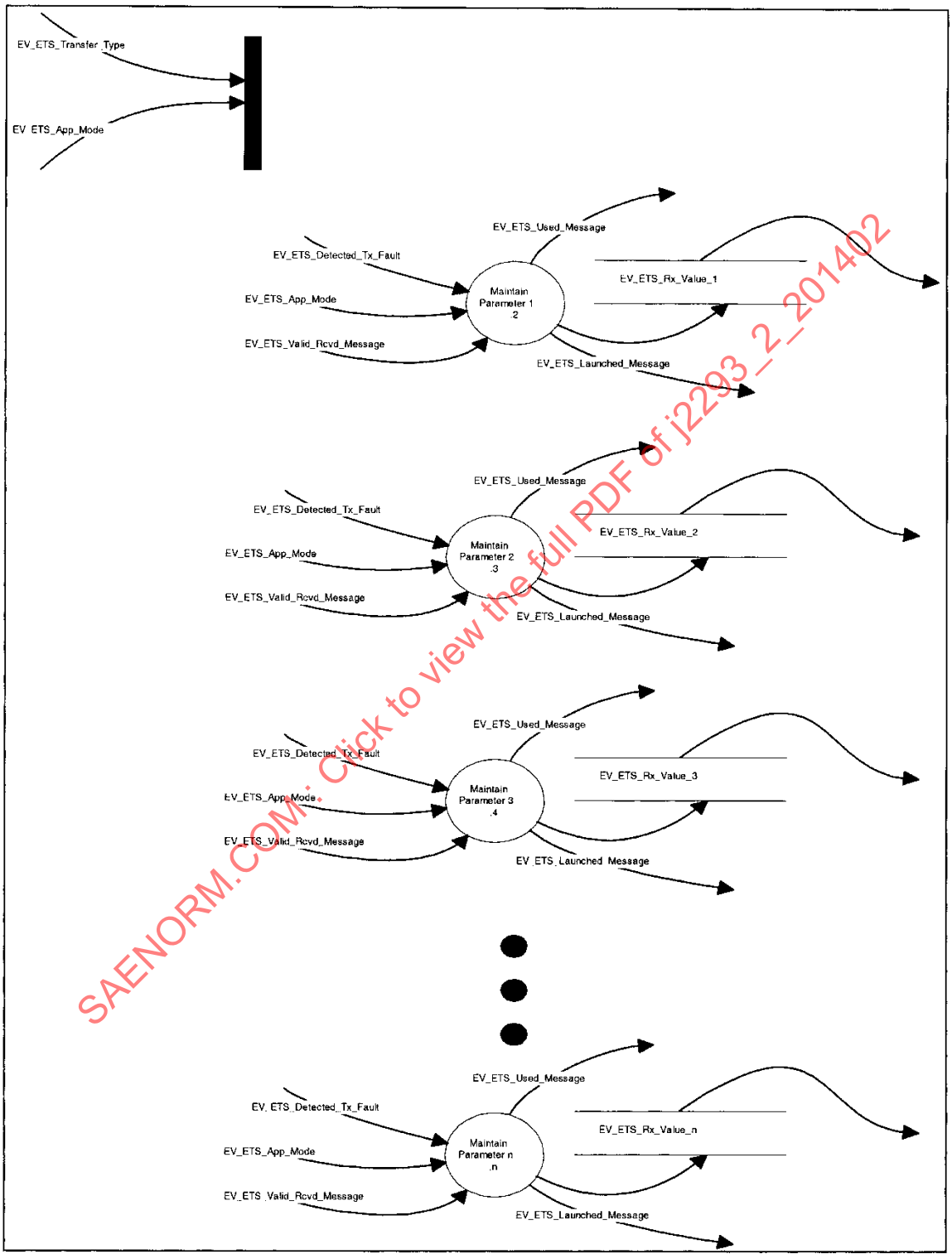

FIGURE 27 - ILLUSTRATION OF MAINTAIN APPLICATION PARAMETERS DFD

1. There shall be one process per message received by the EV portion of the ETS. The messages in a particular system are determined by using Tables 54A, 54B, and 54C.

   Use the following procedure to determine which messages require processes.

   a. Select the columns in Tables 54A, 54B, and 54C which are under the ETS architectures supported.

   b. Every message which is marked with an 'R' under the EV Rx column requires a launch process.

   c. Every message which is marked with an 'O' under the EV Rx column may optionally have a launch process created for them at the manufacturer's discretion.

2. The process for each message shall be functionally equivalent to 6.9 with the following substitutions:

   For each message identified in the previous paragraph, use the following procedure to generate the requirements for the process to maintain the parameters in each message.

   a. For the message this process controls, using Table 56, substitute the dataflows identified in the Dataflows Contained" column into the context flow Rx_Value.

   b. Substitute EV_ETS_App_Mode for the context flow Mode.

   c. Substitute EV_ETS_Valid_Rcvd_Message for the context flow Valid_Rcvd_Message.

   d. Substitute EV_ETS_Used_Message for Used_Message.

   e. Substitute EV_ETS_Detected_Tx_Fault for Transmitter_Fault.

   f. Substitute EV_ETS_Launched_Message for Launched_Message.

   g. For the message this process controls, using Table 55 and Table 48, substitute the values for this message's receipt strategy into the configuration stores in the maintain parameter process template.

6.3.3.2.3.1 C-spec—Maintain EV ETS Application Parameters PAT

a. *Input/Output Flows*

   1. *Flow Name* — *Type*
      EV_ETS_App_Mode — INPUT

   2. *Processes Controlled*—all launch processes contained in 6.1.3.2.3

b. *Initialization*—The following shall be performed once upon activation of this process:

   All processes controlled by this P-spec shall be disabled.

c. *Operation*—The following shall be performed continually while the process which contains this C-spec is active:

   a. Every process in 6.1.3.2.3 shall be controlled via this Process Activation Table.

   b. The process which corresponds to a message parameter shall be enabled for operation when the EV_ETS_App_Mode is equal to one of the states in column labeled Enabled State(s) for that parameter and the message is permitted for the value of Transfer_Type as defined in Tables 54A, 54B, and 54C.

6.3.3.2.4 Detect Application Transmitter Fault in EV

These processes learn the transmitter which are in use by the ETS system and detect faults when any of those transmitters stop transmitting. These processes reset the learning any time there is an application mode change. There are seven detectors to accommodate the following transmitters:

1. Three transmitters from the EV ETS application
2. One LMS transmitter
3. One off-board external interface transmitter
4. One EV external interface transmitter
5. One learned transmitter for future expansion

a. *Input/OutputFlows*

Flow Name	*Type*
EV_ETS_Valid_Rcvd_Message	INPUT
EV_ETS_Used_Message	INPUT
EV_ETS_App_Mode	INPUT
EV_ETS_Detected_Tx_Fault	OUTPUT

b. *Operation*—This process shall perform functionally identical to 5.7 Detect Application Transmitter Fault Template with the substitutions and exceptions which follow.

1. The context flows for 6.7 shall be substituted as defined in Table 22.

TABLE 22 - DETECT APPLICATION FAULT TEMPLATE CONTEXT FLOW SUBSTITUTIONS

Context Flow	Local Substitution Flow
Used_Message	EV_ETS_Used_Message
Detected_Tx_Fault	EV_ETS_Detected_Tx_Fault
Mode	EV_ETS_App_Mode
Valid_Rcvd_Message	EV_ETS_Valid_Rcvd_Message

6.3.3.3 Manage EV ETS Communications

a. *Input/Output Flows*

Flow Name	*Type*
EV_ETS_App_Service_Request	INPUT
EV_ETS_App_Tx_Enabled	OUTPUT
EV_ETS_App_Rx_Enabled	OUTPUT
EV_ETS_Launched_Message	INPUT
EV_ETS_Tx_Data	INPUT
EV_Bus_State	INPUT
EV_ETS_Valid_Rcvd_Message	OUTPUT
Coupling_Type	INPUT
EV_ETS_Transmission_State	OUTPUT

b. *Operation*—This process shall perform functionally identical to 6.6 with the substitutions and exceptions which follow.

1. The context flows for 6.6 shall be substituted as defined in Table 23.

TABLE 23 - MANAGE EV ETS COMMUNICATIONS CONTEXT FLOW SUBSTITUTIONS

Context Flow	Local Substitution Flow
App_Rx_Enabled	EV_ETS_App_Rx_Enabled
App_Service_Request	EV_ETS_App_Service_Request
App_Tx_Enabled	EV_ETS_App_Tx_Enabled
Bus_State	EV_Bus_State
Coupling_Type	Coupling_Type
Launched_Message	EV_ETS_Launched_Message
Location	ONBOARD
Transmission	EV_ETS_Transmission
Tx_Data	EV_ETS_Tx_Data
Valid_Primary_IDs	hexadecimal 70, hexadecimal 71, hexadecimal FE, hexadecimal FF
Valid_Rcvd_Message	EV_ETS_Valid_Rcvd_Message

6.3.4 Transceive EV External ETS Information

This group of processes define the requirements to:

a. Receive messages from the SAE J1850 bus

b. Detect which messages are directed to the external portion of the ETS in the EVSE system

c. Decode those messages to obtain information for ETS dataflows

d. Determine when to update information in other parts of the ETS System

e. Encode ETS information to send messages to the rest of the ETS system

f. Send messages to the rest of the ETS system via the J1850 Bus

g. Provide basic protocols for fault recovery

h. Provide vehicle control of application communication

This process shall perform functionally identical to 6.1.4 with the exception that the value of location in Table 10 shall be ONBOARD.

6.4 Store—Coupling_Type

This store contains the connector configuration for the ETS system.

6.5 Manage Application Slave Nodey Communication Template

6.5.1 Context for Manage Application Slave Node Communication Template

This template is applied throughout this document by first substituting flows from the local DFD in place of context flows shown in Figure 28.

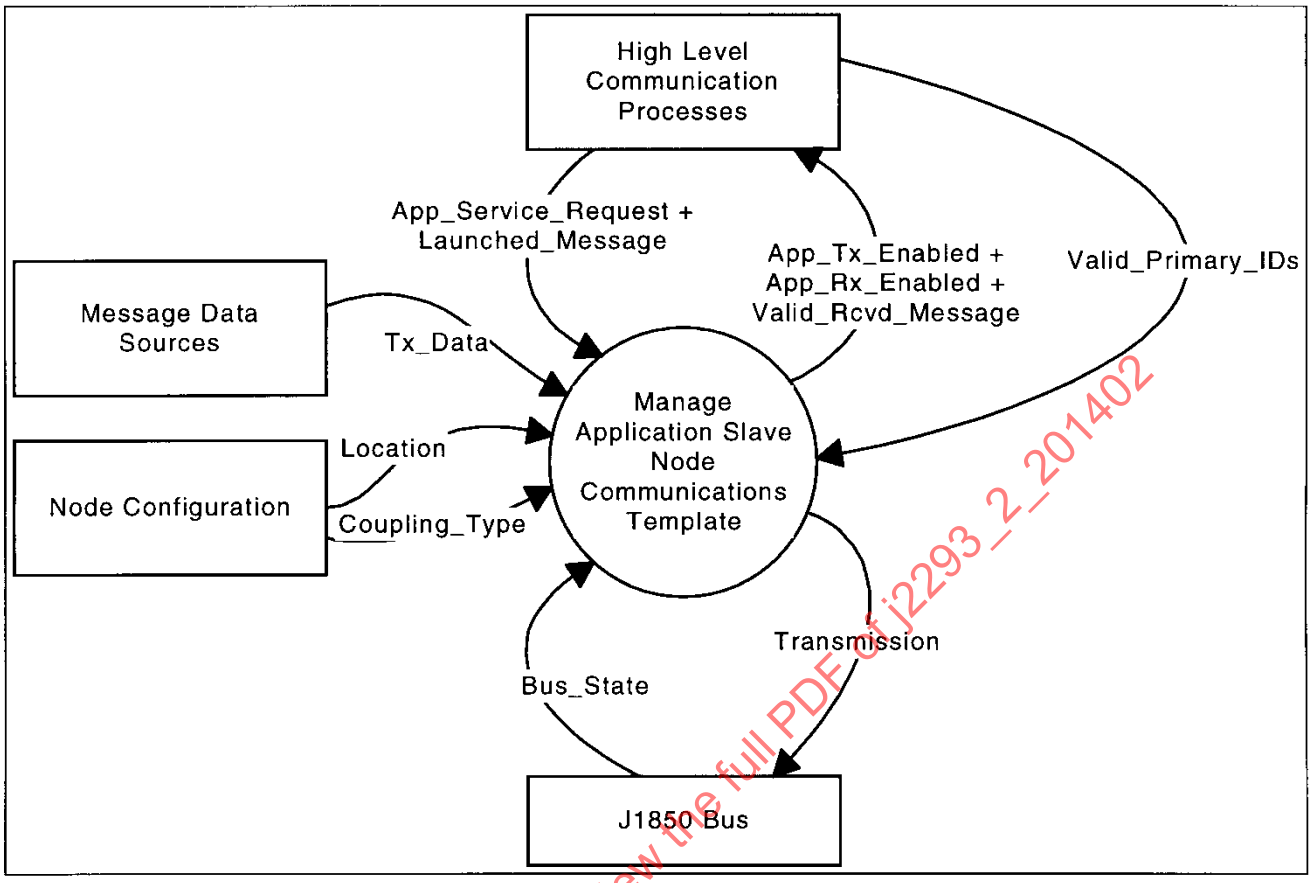

FIGURE 28 - MANAGE APPLICATION SLAVE NODE COMMUNICATION CONTEXT DIAGRAM

6.5.2 Functional Requirements For Manage Application Slave Node Communication Template

The Application Slave Node Template provides a set of requirements which define how a slave node:

a. Receives messages from the J1850 bus

b. Detects which messages are directed to a specific part of an application

c. Decodes those messages to obtain information for the application dataflows

d. Determines when to update information in other parts of the application

e. Encodes application information in this node to send to other parts of the application in different nodes

f. Send information to other parts of an application using messages on the SAE J1850 Bus

g. Support basic fault recovery protocols

h. Determines if it is enabled for communication with a master application node

Figure 29 illustrates the relationship between these processes.

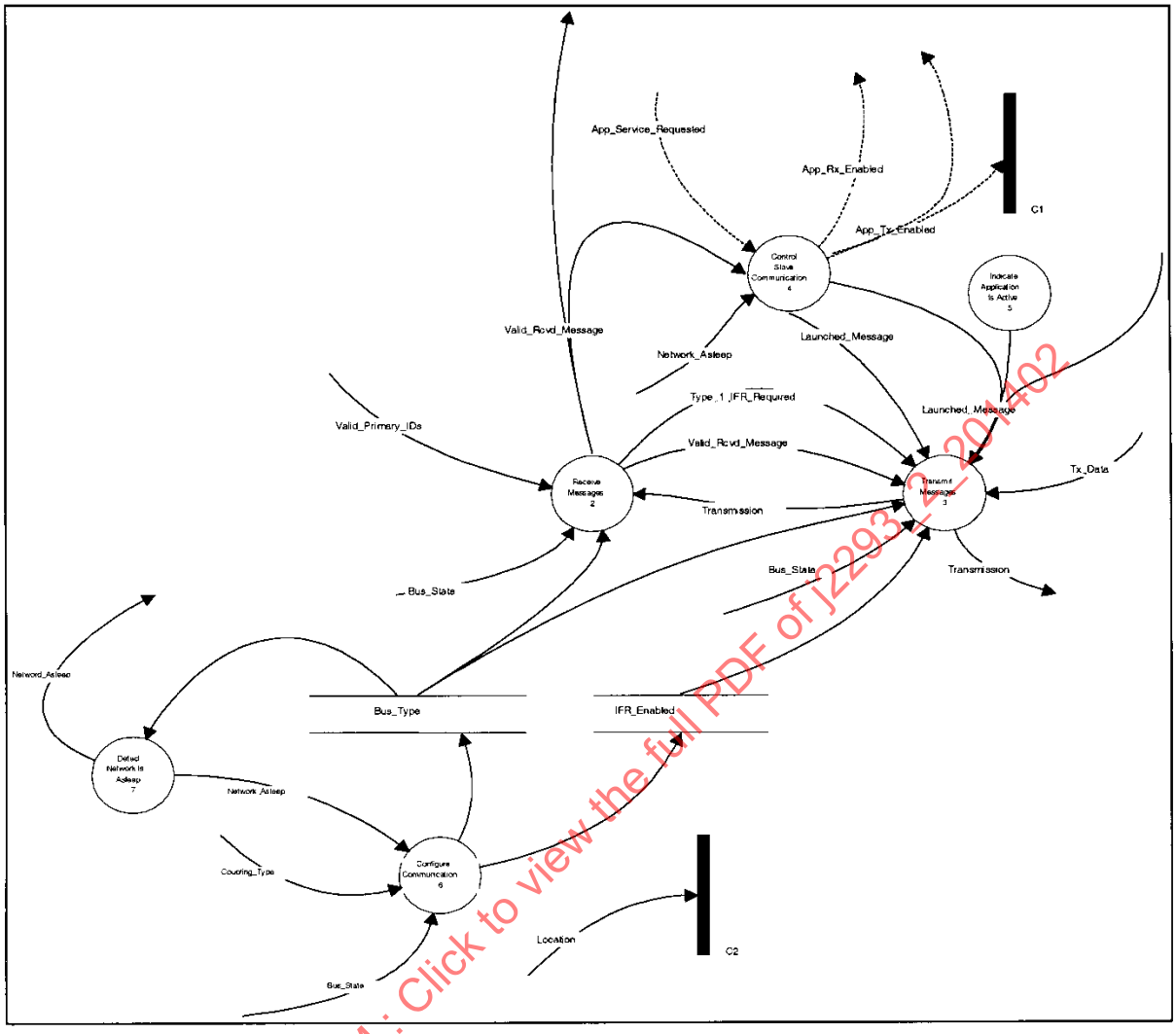

FIGURE 29 - MANAGE APPLICATION SLAVE NODE COMMUNICATION DFD

6.5.2.1 Process Control Requirements

6.5.2.1.1 C-spec—Indicate Application is Active in Node PAT.

a. *Inputs, Outputs, Processes Controlled*.

1. *Flow Name* *Type*
   App_Tx_Is_Enabled INPUT

2. *Process Controlled*—6.5.2.5 P-spec—Indicate Application is Active in Node

b. *Initialization*—None.

c. *Operation*—The following shall be performed continually while the process which contains this C-spec is active:

The process identified in Table 24 shall be enabled and disabled based on the state of the control flow in this table.

TABLE 24 - INDICATE APP IS ACTIVE IN NODE PAT

Control Flow App_Tx_Enable	Process 6.5.2.5 P-spec—Indicate Application is Active in Node
FALSE	DISABLED
TRUE	ENABLED

6.5.2.1.2 C-spec—Configure Communication PAT.

a. *Inputs, Outputs, Processes Controlled*.

1. *Flow Name* *Type*
   Location INPUT

2. *Process Controlled*—6.5.2.6 P-spec—Configure Communication

b. *Initialization*—6.5.2.6 P-spec—Configure Communication shall be disabled.

c. *Operation*—The following shall be performed continually while the process which contains this C-spec is active:

The process identified in Table 25 shall be enabled and disabled based on the state of the control flow in this table.

TABLE 25 - INDICATE APP IS ACTIVE IN NODE PAT

Control Flow Location	Process 6.5.2.6 P-spec—Configure Communication
ONBOARD	DISABLED
OFF-BOARD	ENABLED

6.5.2.2 Receive Messages

This group of processes performs the following functions:

a. Translates the stream of symbols on the SAE J1850 bus media in use into a message less any In Frame Response (IFR)

b. Accepts only messages with valid CRC bytes

c. Accepts only messages which are encode for compatibility with SAE J2293 thus allowing compatibility with unconsolidated headers

d. Decodes the message for its operator and parameter name based on SAE J2178 definitions

e. Determines when an IFR is required

Figure 30 illustrates the relationship between these processes.

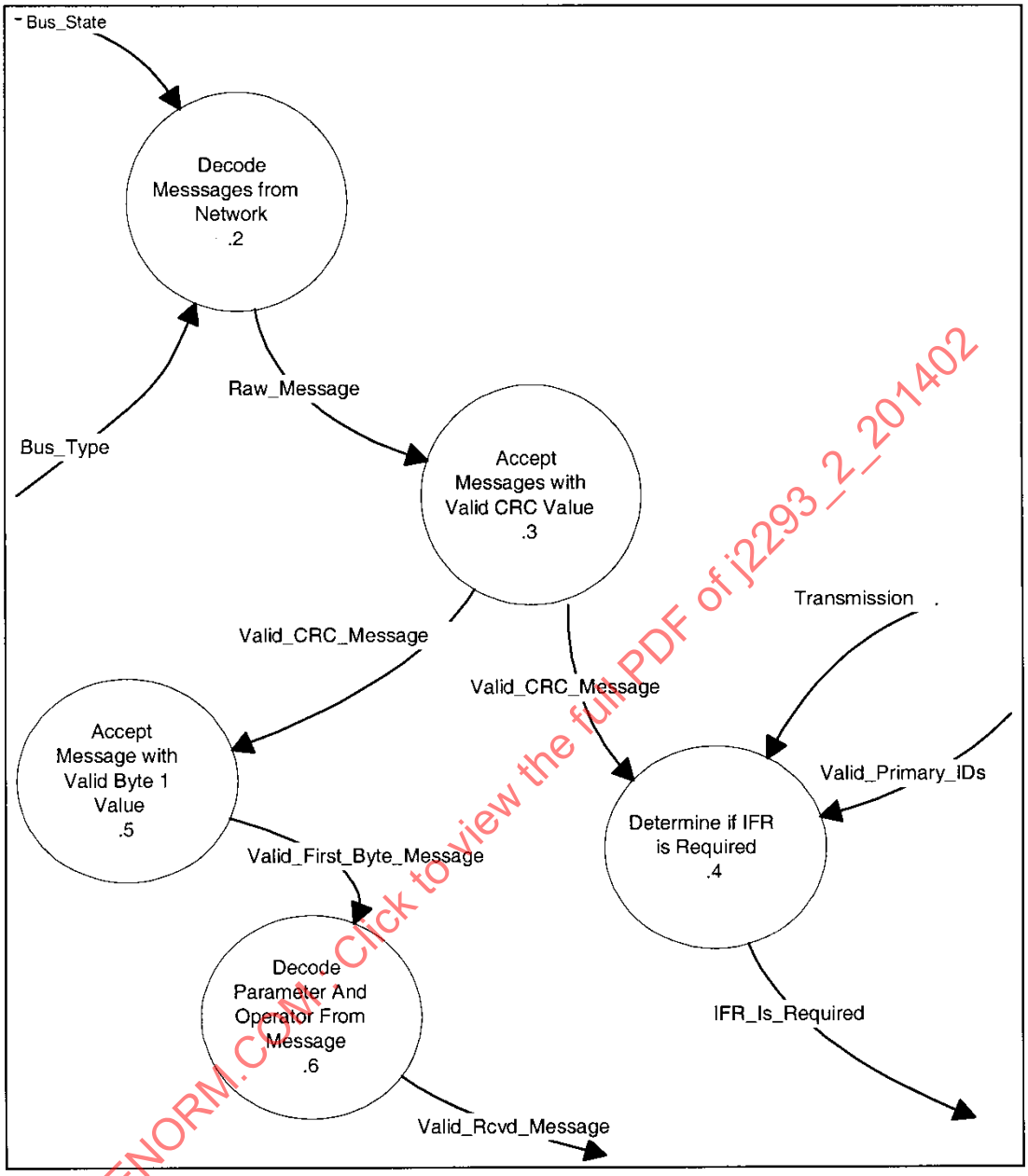

FIGURE 30 - RECEIVE MESSAGES DFD

6.5.2.2.1 Process Control Requirements

None.

6.5.2.2.2 P-spec—Decode Messages from the Network

This process decodes the symbols on the SAE J1850 network and assembles the information into messages.

a. *Input/Output Flows*

Flow Name	*Type*
Dual_Wire_Bus_State	INPUT
Single_Wire_Bus_State	INPUT
Bit_Encoding	INPUT
Data_Rate	INPUT
Bus_Media	INPUT
Raw_Message	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—The following shall be performed continually while this process is active:

1. If Bus_Media is DUAL_WIRE then perform the following:
   Decode the state of the Dual_Wire_Bus_State versus time to determine the sequence of symbols on the network. The symbols shall be decoded based on Bit_Encoding and Data_Rate. The symbol timing is defined in SAE J1850. In the event that the bus is operating in a SAE J1850 defined fault mode, the symbols shall be decoded properly.
   Assemble the sequence of symbols into a message less any In Frame Response. Message and In Frame Response are defined in SAE J1850. The value of this message shall be assigned to Raw_Message.
   Issue Raw_Message.

2. If Bus_Media is SINGLE_WIRE, then perform the following:
   Decode the state of the Single_Wire_Bus_State versus time to determine the sequence of symbols on the network. The symbols shall be decoded based on Bit_Encoding and Data_Rate. The symbol timing is defined in SAE J1850.
   Assemble the sequence of symbols into a message less any In Frame Response. Message and In Frame Response are defined in SAE J1850. The value of this message shall be assigned to Raw_Message.
   Issue Raw_Message.

6.5.2.2.3 P-spec—Accept Message with Valid CRC Value

This process decodes the last byte of a message and determines if the CRC is valid as defined in SAE J1850. If the message has an invalid CRC it is rejected and not used for further processing.

a. *Input/Output Flows*

Flow Name	*Type*
Raw_Message	INPUT
Valid_CRC_Message	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—While this process is active, the following shall be performed upon receipt of Raw_Message:

If the last byte of Raw_Message has a value which indicates a valid CRC as defined in SAE J1850, then:

Issue Valid_CRC_Message with the value of Raw_Message.

6.5.2.2.4 P-spec—Determine if IFR Required

This process determines if an In Frame Response (IFR) is required to be generated. An IFR is only generated if a message has a valid CRC, has a primary ID used in the ETS system, and was not generated in this node.

a. *Input/Output Flows*

Flow Name	*Type*
Transmission	INPUT
Valid_CRC_Message	INPUT
Valid_Primary_IDs	INPUT
Type_1_IFR_Required	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—While this process is active, the following shall be performed upon receipt of Valid_CRC_Message:

If the K Bit, as defined in SAE J1850 in byte 1 of Valid_CRC_Message has a value of 0 and the Primary ID of the message is one of the Primary_IDs in Valid_Primary_IDs and the message is not equal to the message encoded in Transmission then:

Issue Type_1_IFR_Required with the value of TRUE.

6.5.2.2.5 P-spec—Accept Message with Valid Byte 1 Value

SAE J1850 and SAE J2178 allow for unconsolidated header messages and consolidated header messages. Different parameters may be encoded using these two techniques which result in identical messages. This is a potential source of problems for off-board systems which will connect to vehicle implementing both encoding techniques. This requirement for off-board systems allows compatibility with both encoding techniques. By forcing the off-board systems to reject messages which do not have a specific first byte value, unconsolidated header messages are rejected. After filtering by this process, only valid three byte header messages are available for subsequent processing in the off-board system.

a. *Input/Output Flows*

Flow Name	*Type*
Valid_CRC_Message	INPUT
Valid_First_Byte_Message	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—While this process is active, the following shall be performed upon receipt of Valid_CRC_Message:

If the value of Byte 1 of Valid_CRC_Message is equal to hexadecimal 20, 21, 28 or 29 then:

Issue Valid_First_Byte_Message with the value of Valid_CRC_Message.

6.5.2.2.6 P-spec—Decode Parameter and Operator from Message

All ETS messages are encoded using three byte consolidated headers. This process decodes the first four bytes of a message to determine the parameter name and parameter operator encoded in the message header. A convention of attaching the parameter name and parameter operator to the message as it is passed among processes is used to simplify the description of processing requirements on specific messages.

a. *Input/Output Flows*

Flow Name	Type
Valid_First_Byte_Message	INPUT
Valid_Rcvd_Message	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—While this process is active, the following shall be performed upon receipt of Valid_First_Byte_Message:

1. Valid_First_Byte_Message shall be decoded as required in SAE J2178 and SAE J1850 to decode the parameter name encoded in the message.

   Assign this name to Parameter_Name.

2. Valid_First_Byte_Message shall be decoded as required in Table 26.

   Assign this operator to Operator.

3. If the Parameter Name is a valid SAE J2178 parameter name and the operator is one of the operators defined in Table 26, then:

   Issue Valid_Rcvd_Message with the value of Valid_First_Byte_Message + Operator + Parameter_ Name.

TABLE 26 - FUNCTION BIT DECODING TABLE

Input K Bit	Input Y Bit	Input Z1 Bit	Input Z0 Bit	Input W Bit	Input Q Bit	Input C Bit	Output Operator
1	0	0	0	0	Don't Care	0	LD
0	0	0	0	0	Don't Care	0	LD
1	0	0	1	1	0	0	RQCV
0	0	0	0	1	0	1	RQCV
1	0	0	0	1	Don't Care	0	RPT
0	0	0	0	1	Don't Care	0	RPT
1	0	0	1	1	0	0	REQ
0	0	0	1	1	1	1	REQ

6.5.2.3 Transmit Messages

This group of processes provides the functions of:

a. Assigning Primary ID's, Secondary ID's, and function bits to a message

b. Managing the network to allow nodes sufficient time to wakeup prior to communication traffic

c. Queue messages

d. Set the order that queued messages are submitted for transmission

e. Manage retransmission to prevent low priority messages from clogging the transmission queue

f. Encode data with a message

g. Translate the message into a stream of symbols on the SAE J1850 bus media in use

h. Confirm transmission

i. Provide In Frame Responses (IFR)

Figure 31 illustrates the relationship between these processes.

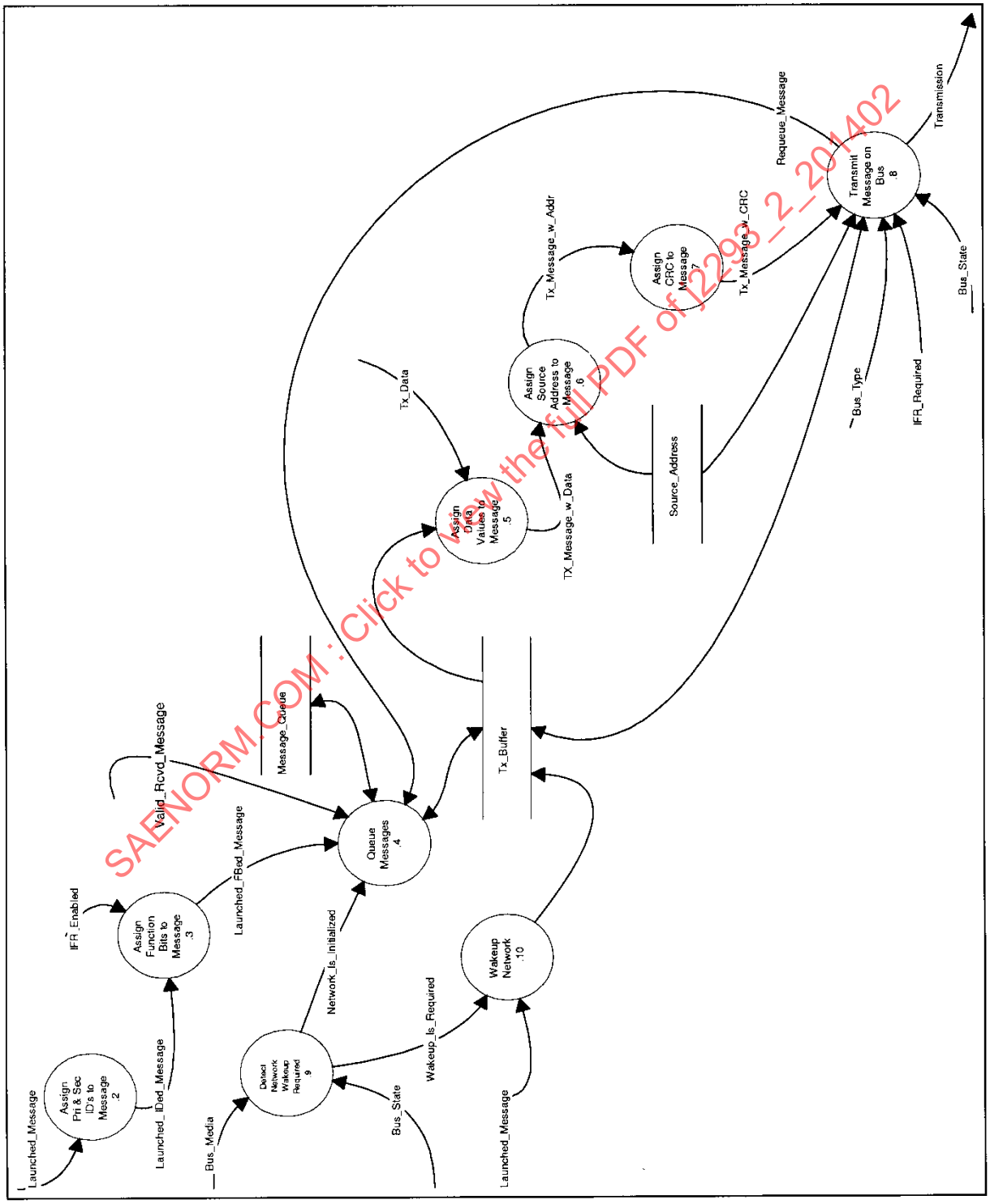

FIGURE 31 - TRANSMIT MESSAGES DFD

6.5.2.3.1 Process Control Requirements

None.

6.5.2.3.2 P-spec—Assign Primary And Secondary ID's to Message

This process assigns a value to the primary and secondary ID bits in the message based on the message parameter name.

a. *Input/Output Flows*

Flow Name	*Type*
Launched Message	INPUT
Launched_IDed_Message	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—While this process is active, the following shall be performed on receipt of Launched_ Message:

1. Based on the value of Parameter Name in Launched_Message assign the Primary ID bits in Byte 2 of Launched_Message using the definitions in SAE J2178 for that parameter.

2. Based on the value of Parameter Name in Launched_Message assign the Secondary ID bits in Byte 4 of Launched_Message using the definitions in SAE J2178 for that parameter.

3. Issue Launched_IDed_Message with the value of Launched_Message.

6.5.2.3.3 P-spec—Assign Function Bits to Message

This process assigns a value to the function bits in the message based on the operator in this message.

a. *Input/Output Flows*

Flow Name	*Type*
Launched_IDed_Message	INPUT
IFR_Enabled	INPUT
Launched_FBed_Message	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—While this process is active, the following shall be performed on receipt of Launched_IDed_Message:

1. Based on the value of Operator in Launched_IDed_Message and the value of IFR_Enabled, use to encode the value of the function bits. These function bits shall be assigned to Launched_IDed_Message data bytes as defined in SAE J2178 and SAE J1850.

TABLE 27 - FUNCTION BIT ENCODING

Input Operator	Input IFR_Enabled	Output K Bit	Output Y Bit	Output Z1 Bit	Output Z0 Bit	Output W Bit	Output Q Bit	Output C Bit
LD	FALSE	1	0	0	0	0	n/a	0
LD	TRUE	0	0	0	0	0	n/a	0
RQCV	FALSE	1	0	0	1	1	0	0
RQCV	TRUE	0	0	0	0	1	0	1
RPT	FALSE	1	0	0	0	1	n/a	0
RPT	TRUE	0	0	0	0	1	n/a	0
REQ	FALSE	1	0	0	1	0	0	0
REQ	TRUE	0	0	0	1	1	1	1

2. The upper nibble in byte 1 of Launched_IDed_Message shall be assigned the values of 0010 respectively. These function bits shall be assigned to the bits in the byte 1 as defined in SAE J2178 and SAE J1850.

3. Issue Launched_FBed_Message with the value of Launched_IDed_Message.

6.5.2.3.4 P-spec—Manage Message Queue

This process holds messages which have been launched until the network is initialized and the transmission processes are able to transmit on the bus.

This process maintains the queue so that any message requests (REQ or RQCV) are dequeued if the corresponding LD or RPT is received prior to launch of the request.

This process maintains the queue so that duplicate messages are not allowed. Messages are identical if their parameter name, operator, and any assigned data are identical. For example, the messages

"App Comm State", LD, App Type=SAE, App_ID=2293 ETS, App_State=Unassigned
"App Comm State", LD, App Type=SAE, App_ID=Security, App State=Unassigned

are unique messages due to the values assigned to App_ID and App_State. When submitted to the queue, these two messages would be maintained for transmission.

To describe the interaction between this process and the processes which transmit a message on the SAE J1850 bus, the Tx_Buffer is used as a flag. When this store is empty, it indicates that the transmission processes are ready to accept a message for transmission. While this store contains a message, it indicates that the transmission processes are attempting to transmit that message and are not ready to accept another message.

This convention has been chosen to avoid the problem of describing the requirements to interact with any specific manufacturer's protocol hardware.

a. *Input/Output Flows*

Flow Name	*Type*
Valid_Rcvd_Message	INPUT
Requeue_Message	INPUT
Launched_FBed_Message	INPUT
Network_Is_Initialized	INPUT
Message_Queue	INPUT/OUTPUT
Tx_Buffer	INPUT/OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*

1. While this process is active, the following shall be performed on receipt of Launched_FBed_ Message or Requeue_Message:

   If there are not any messages in Queue which have the same Parameter Name, Operator and data as Launched_FBed_Message then store this message in Queue.

2. While this process is active, the following shall be performed on Receipt of Valid_Rcvd_Message:

   If Valid_Rcvd_Message has an operator of LD and there is a message in Message_Queue which has the same Parameter Name as Valid_Rcvd_Message and an Operator of RQCV then remove the RQCV message from Message_Queue.

   If Valid_Rcvd_Message has an operator of RPT and there is a message in Message_Queue which has the same Parameter Name as Valid_Rcvd_Message and an Operator of REQ then remove the REQ message from Message_Queue.

3. The following shall be performed continually while this process is active:

   If Tx_Buffer is Empty and Message_Queue has one or more message stored and Network_Initialized has a value of TRUE, then remove a message from Message_Queue and place in Tx_Buffer. If possible, the highest priority message, as defined in SAE J1850 should be moved to Tx_Buffer.

6.5.2.3.5 P-spec—Assign Data Values to Message

This process assigns data to a message immediately prior to any attempt to transmit the message on the network. Some messages have data which is assigned prior to this process. For messages such as these, only data fields in the message which are not assigned a value are updated. This is done to allow multiple instances of messages such as "App Comm State" to be placed in the queue. For this message, the two data fields App ID and App Type may have been assigned by the process which launched the message. The data field App State may be unassigned by the time the message reaches this process. This process will leave App ID and App Type the value they were when launched. Since App State is unassigned, this process will assign data to this field.

a. *Input/Output Flows*

Flow Name	*Type*
Tx_Buffer	INPUT
Tx_Data	INPUT
Tx_Message_w_Data	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process—None.

c. *Operation*—The following shall be performed continually while this process is active:

If Tx_Buffer has an Operator of RPT or LD then based on the parameter encoding defined in SAE J2178 for this message. If no data has been previously assigned to any portion of the data field in Tx_Buffer, encode the appropriate value in Tx_Data to the appropriate bytes in the message.
Assign this result to Tx_Message_w_Data.

6.5.2.3.6 P-spec: Assign Source Address to Message

This process assigns the transmitter source address to the third byte in the message prior to any attempt to transmit on the network.

a. *Input/Output Flows*

Flow Name	*Type*
Source_Address	INPUT
Tx_Message_w_Data	INPUT
Tx_Message_w_Addr	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—The following shall be performed continually while this process is active:

Assign Tx_Message_w_Addr the value of Tx_Message_w_Data and assign byte 3 in Tx_Message_w_ Addr the value of Source_Address.

6.5.2.3.7 P-spec—Assign CRC to Message

This process assigns the CRC value to the message immediately prior to any attempt to transmit.

a. *Input/Output Flows*

Flow Name	*Type*
Tx_Message_w_Addr	INPUT
Tx_Message_w_CRC	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—The following shall be performed continually while this process is active:

1. Assign Tx_Message_w_CRC the value of Tx_Message_w_Addr

2. Assign the byte following the last data byte in Tx_Message_W_CRC an 8 bit CRC calculated as required in SAE J1850.

6.5.2.3.8 P-spec—Transmit Message on Bus

This process performs the function of transmitting a message on the network media in use, per the requirements in SAE J1850 for arbitration, bit encoding, data rate, etc.

This process also determines when a message is blocked sufficiently by other bus traffic to require message requeueing. In the event a message can not win arbitration within a sufficient period of time, that message is removed from the transmit buffer and resubmitted to the queue.

This process also transmits an In Frame Response on Dual Wire SAE J1850 implementations. In Frame Response is not supported for Single Wire implementations.

a. *Input/Output Flows*

Flow Name	*Type*
Bit_Encoding	INPUT
Bus_Media	INPUT
Data_Rate	INPUT
Type_1_IFR_Required	INPUT
Tx_Message_w_CRC	INPUT
Dual_Wire_Bus_State	INPUT
Single_Wire_Bus_State	INPUT
Requeue_Message	OUTPUT
Tx_Buffer	INPUT/OUTPUT
Dual_Wire_Transmission	OUTPUT
Single_Wire_Transmission	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*

1. While this process is active, the following shall be performed upon receipt of Type_1_IFR_Required:

   If Bus_Media is DUAL_WIRE then,

   Issue a string of symbols to Dual_Wire_Transmission, to provide the In Frame Response with a value equal to Source_Address as defined in SAE J1850 until either the address is transmitted or the EOF of the message.

2. While this process is active, the following shall be performed when the stream of values on Dual_Wire_Bus_State forms a message, less any IFR:

NOTE: Some hardware implementations of dual wire J1850 provide for two (2) automatic retransmissions of messages if an IFR is not received in response to a transmission attempt

   If Bus_Media is DUAL_WIRE

   and Tx_Message_w_CRC is not NULL
   and the symbols received on Dual_Wire_Media are equal to the attempted transmission, then
   Clear Tx_Buffer.

   If Bus_Media is DUAL_WIRE
   and Tx_Message_w_CRC is not NULL
   and the symbols received on Dual_Wire_Media are not equal to the attempted transmission
   and Tx_Message_w_CRC has been the same value continuously for the preceding 20 msec (+20 msec, -0 msec) then,
   Issue Requeue_Message with the value of EVSE_Tx_Buffer.
   Clear Tx_Buffer.

3. While this process is active, the following shall be performed while the stream of values on Dual_Wire_Bus_State indicates that the bus is idle:

   If Bus_Media is DUAL_WIRE
   and Tx_Message_w_CRC does not have a value of NULL then,
   per SAE J1850, generate a stream of symbols which encode the message in Tx_Buffer using Data_Rate and Bit_Encoding and issue them to Dual_Wire_Transmission until the either arbitration is lost or the message is transmitted.

4. While this process is active, the following shall be performed when the stream of values on Single_Wire_Bus_State forms a message, less any IFR:

   If Bus_Media is SINGLE_WIRE
   and Tx_Message_w_CRC is not NULL
   and the symbols received on Single_Wire_Media are equal to the attempted transmission, then
   Clear Tx_Buffer.

   (The following requirement should be implemented, however this is not required for SAE J2293 compatibility)

   If Bus_Media is SINGLE_WIRE
   and Tx_Message_w_CRC is not NULL
   and the symbols received on Single_Wire_Media are not equal to the attempted transmission
   and Tx_Message_w_CRC has been the same value continuously for the preceding 20 msec (+20 msec, -0 msec) then,
   Issue Requeue_Message with the value of Tx_Buffer.
   Clear Tx_Buffer.

5. While this process is active, the following shall be performed while the stream of values on Single_Wire_Bus_State indicates that the bus is idle:

   If Bus_Media is SINGLE_WIRE
   and Tx_Message_w_CRC does not have a value of NULL then,
   per SAE J1850, generate a stream of symbols which encode the message in Tx_Buffer using Data_Rate and Bit_Encoding and issue them to Single_Wire_Transmission until the either arbitration is lost or the message is transmitted.

6.5.2.3.9 P-spec—Detect Network Wakeup Required

This process monitors network activity on the network media which is currently in use in the EVSE and determines when the network requires a wakeup message prior to data message transmissions. This process also determines when the appropriate period of time has passed for network initialization to complete. While the network is initializing, transmission of any message besides the wakeup messages is prevented.

Figure 32 provides a simplified illustration of the function performed by this process.

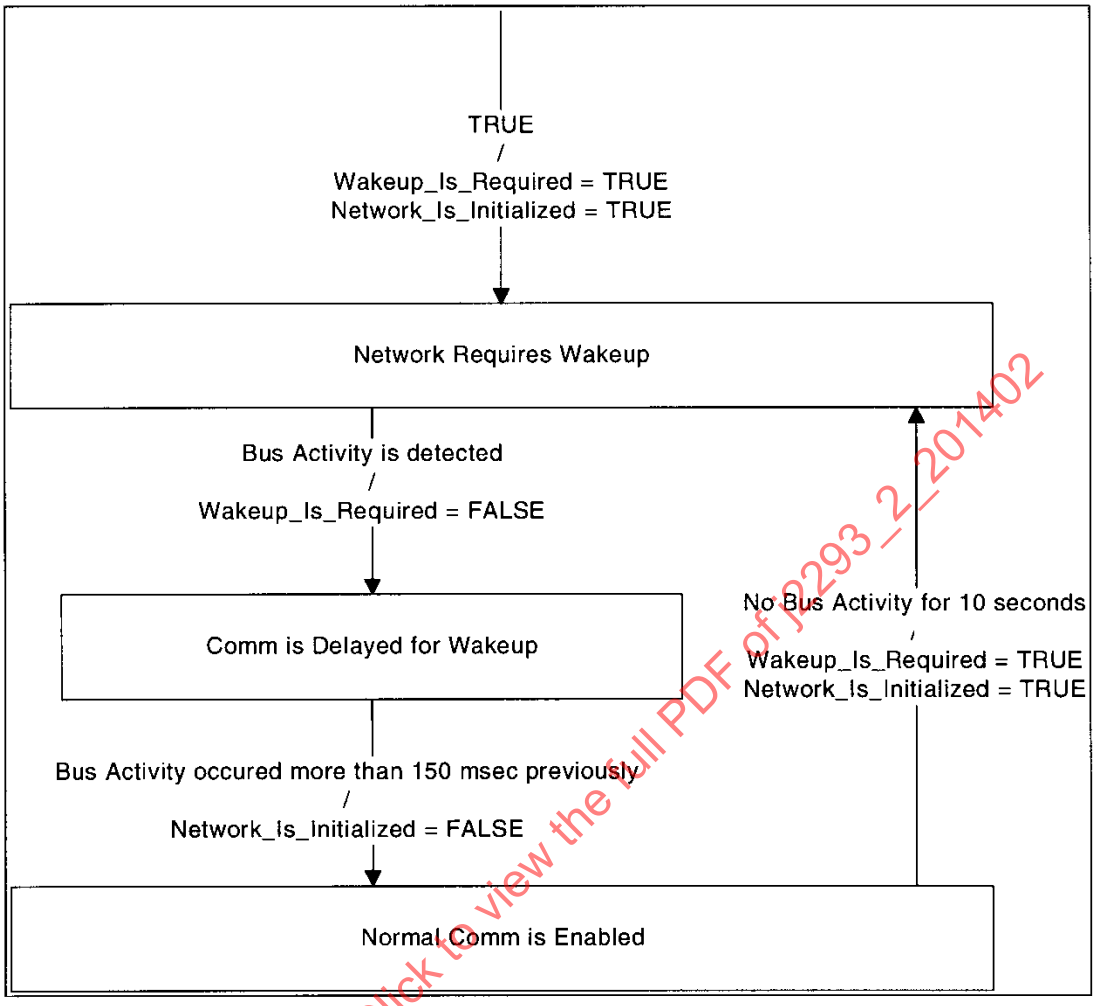

FIGURE 32 - ILLUSTRATION OF NETWORK WAKEUP STD

a. *Input/Output Flows*

Flow Name	*Type*
Single_Wire_Bus_State	INPUT
Dual_Wire_Bus_State	INPUT
Bus_Media	INPUT
Wakeup_Is_Required	OUTPUT
Network_Is_Initialized	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

1. Wakeup_Is_Required shall be assigned the value of TRUE.

2. Network _Initialized shall be assigned the value of FALSE.

c. *Operation*—The following shall be performed continually while this process is active:

1. If Bus_Media is SINGLE_WIRE and Single_Wire_Bus_State has continuously had a value of PASSIVE for the previous 10 seconds (± second) then

   Assign Wakeup_Is_Required the value of TRUE,
   otherwise,
   If Bus_Media is DUAL_WIRE and Dual_Wire_Bus_State has continuously had a value of PASSIVE for the previous 10 seconds (± 1 second) then
   Assign Wakeup_Is_Required the value of TRUE,
   otherwise,
   Assign Wakeup_Is_Required the value of FALSE.

2. If Wakeup_Is_Required has been TRUE at any time during the last 150 msec (±10 msec) then

   Assign Network_Initialized the value of FALSE,
   otherwise,
   Assign Network_Initialized the value of TRUE.

6.5.2.3.10 P-spec—Wakeup Network

a. *Input/Output Flows*

Flow Name	*Type*
Launched_Message	INPUT
Wakeup_Is_Required	INPUT
Tx_Buffer	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—While this process is active, the following shall be performed on receipt of Launched_Message:

If Launched_Message does not have a Parameter Name of “Network Wakeup” and Wakeup_Is_Required has a value of TRUE then perform the following in sequence:

Create a message with the Operator of “RPT”.
Assign to that messagethe Parameter Name of “Network Wakeup”.
Assign the Primary and Secondary ID of the message per the requirements of 6.5.2.3.2.
Assign the function bits of the message per the requirements in 6.5.2.3.3.
Write this message to EVSE_Tx_Buffer.

6.5.2.3.11 Store—Message_Queue

This store is used to maintain the messages which are waiting for transmission. This store shall be large enough to accommodate all messages which may be pending launch at any time in this node.

6.5.2.3.12 Store—Tx_Buffer

This store holds one message which is pending for transmission. It is used to trigger an attempt to transmit. When this store is cleared or has a value of EMPTY, it indicates that the 6.5.2.3.8 process is ready to attempt to transmit information on the network.

6.5.2.3.13 Store—Source_Address

This store holds the SAE J1850 Source Address of the functional node. This source address is unique to the ETS Network and to each physical node on the network. It is not duplicated by any other nodes on the ETS Network.

a. If the node is in the EVSE, the Source_Address store shall be assigned a value from using the EVSEs supported architectures as the selection criteria. See Table 28.

TABLE 28 - EVSE SOURCE ADDRESS SELECTION

Input Architecture(s) Supported	Output Valid Source ID's
AC	Hexadecimal C8
Inductive	Hexadecimal C8, or Hexadecimal C9
DC	Hexadecimal C8, Hexadecimal C9, or Hexadecimal CA
AC & Inductive	Hexadecimal C8
AC & Inductive & DC	Hexadecimal C8
AC & D	Hexadecimal C8
Inductive & DC	Hexadecimal C8, or Hexadecimal C9

b. If the node is in the EV, the Source_Address store shall be assigned a value from Table 29 using the EVs supported architectures as the selection criteria.

TABLE 29 - EVSE SOURCE ADDRESS SELECTION

Input Architecture(s) Supported	Output Valid Source ID's
AC	Hexadecimal C9, Hexadecimal CA, or Hexadecimal CB
Inductive	Hexadecimal CA, or Hexadecimal CB
DC	Hexadecimal CB
AC & Inductive	Hexadecimal C9, Hexadecimal CA, or Hexadecimal CB
AC & Inductive & DC	Hexadecimal C9, Hexadecimal CA, or Hexadecimal CB
AC & DC	Hexadecimal C9, Hexadecimal CA, or Hexadecimal CB
Inductive & DC	Hexadecimal CA, or Hexadecimal CB

c. This store shall remain constant while the Network_Asleep is FALSE.

d. This store shall have a value which is unique among all other nodes in the ETS network.

6.5.2.4 Control Slave Application Communication

This group of processes provides a means for the master portion of the application to control the slaved portion of the application. This convention for controlling distributed applications is applicable to system which has components which are designed independently and may be added outside of the control of the OEM. These processes allow the following aspects of the slaved portion of an application to be controlled by the master application node:

a. When the slave portion of the application is allowed to monitor the network

b. When the slave portion of the application is allowed to transmit on the network

c. When the slave portion of the application can inform the master node of its need to be serviced

Controlling when the slave portion of an application monitors the network is important if the network has normal or diagnostic modes which may potentially confuse the application or use messages which are used by the application. Allowing the master node to enable and disable off-board message monitoring prevents potential problems while retaining maximum flexibility for the vehicle network designer.

Controlling when a slave portion of an application may transmit on the network allows the master node to only enable those applications which the network was designed to accommodate. As other applications are designed to use the network, this provides a means for networks not designed to use an applications to maintain interoperability.

Controlling when a slave portion of an application can inform the master node of its need to be serviced allows applications to wakeup the vehicle and request service from other parts of the application to perform a function. Since networks will be designed which may not accommodate particular applications, there is a means to disable these service requests. This allows conservation of network bandwidth and prevents potential problems with unnecessarily waking the network.

Figure 33 provides an illustration of the relationship between these processes.

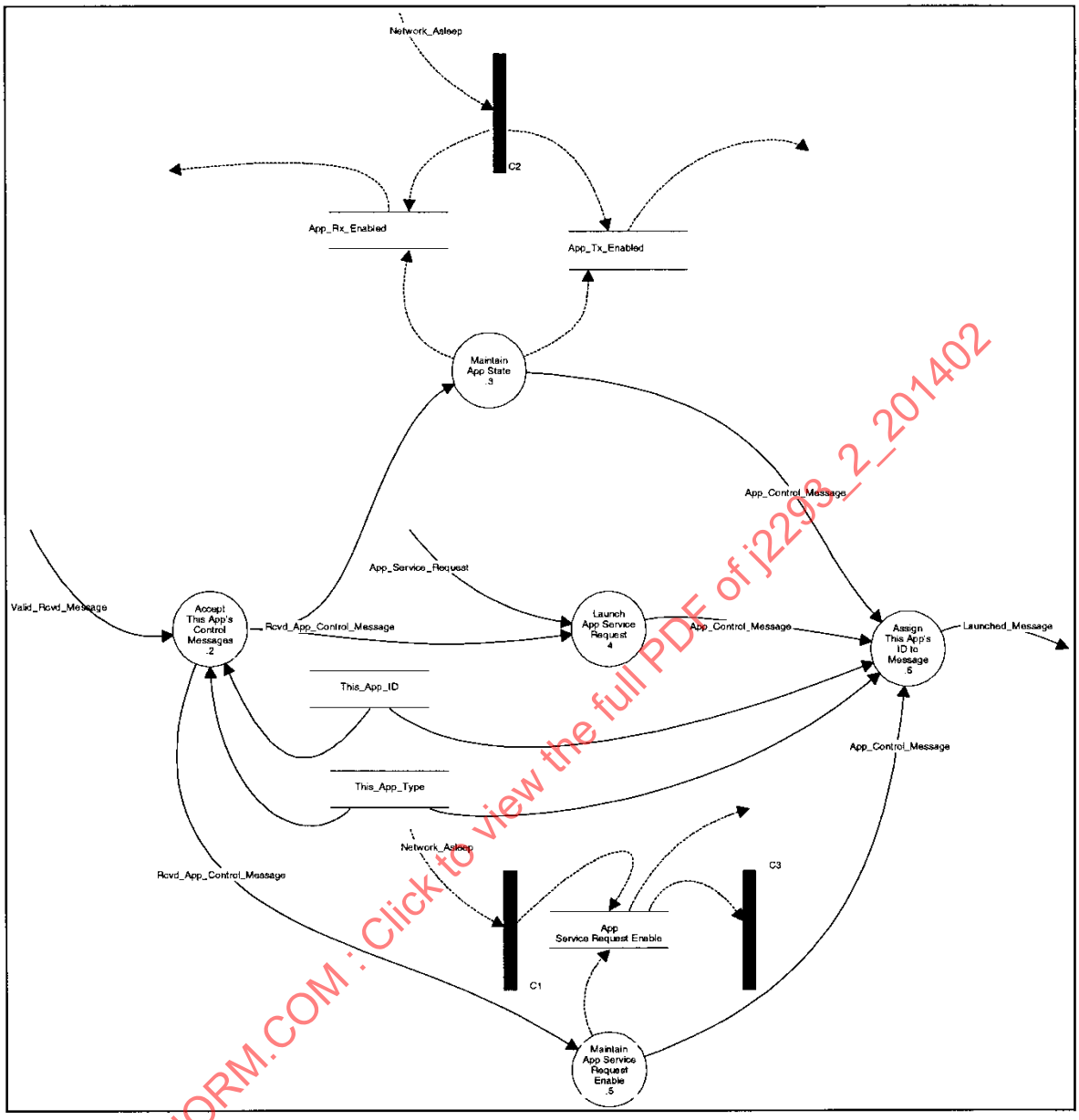

FIGURE 33 - CONTROL APPLICATION SLAVE COMMUNICATION DFD

6.5.2.4.1 Process Control Requirements

6.5.2.4.1.1 C-spec—Reset App_Service_Request_Enable STD

a. *Input/Output Flows*

Flow Name	*Type*
Network_Asleep	INPUT
App_Service_Request_Enable	OUTPUT

b. *Operation*—The following shall be performed continually while the process which contains this C-spec is active:

This process shall behave functionally equivalent to the State Transition Diagram shown in Figure 34.

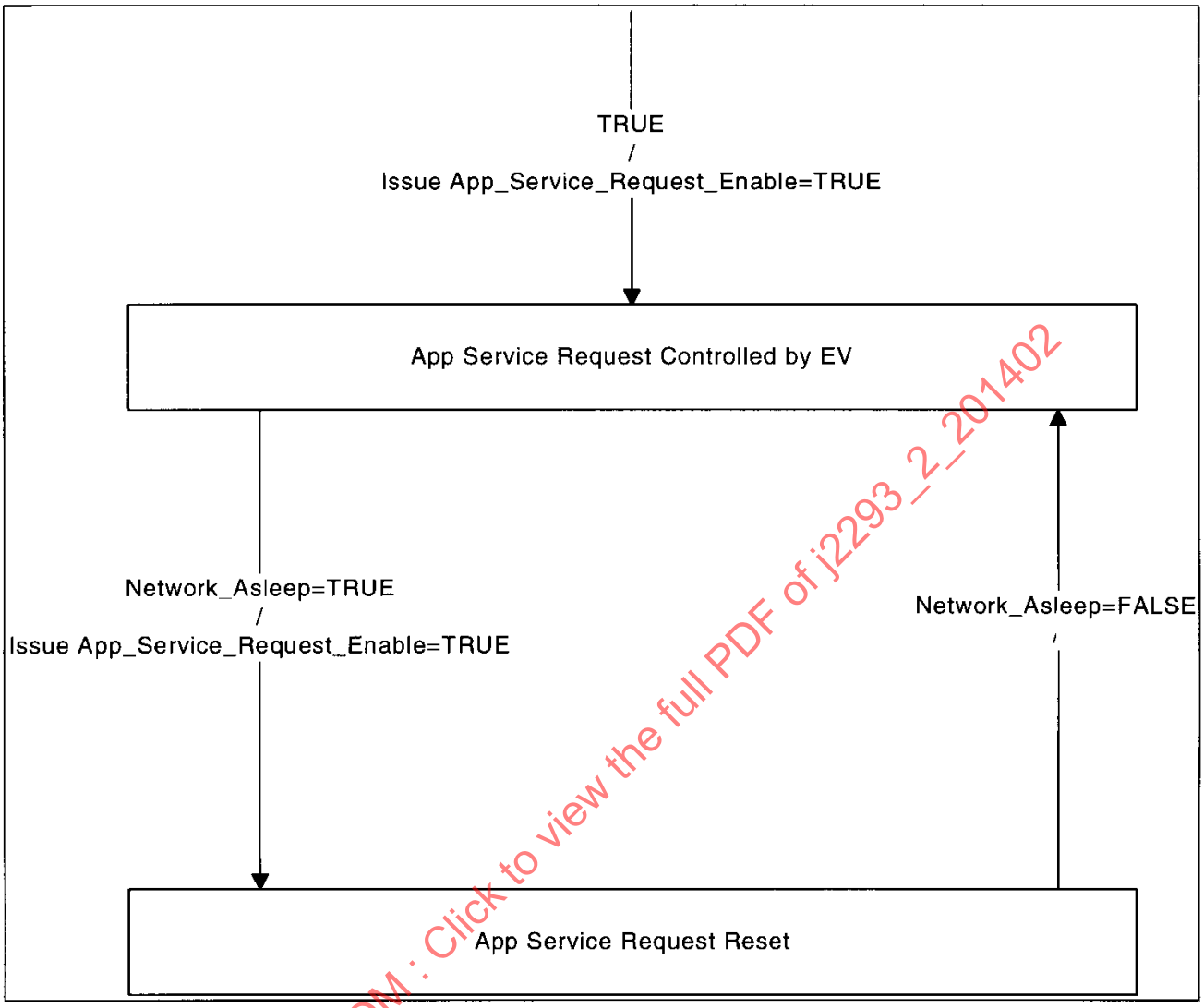

FIGURE 34 - RESET APP_SERVICE_REQUEST_ENABLE STD

6.5.2.4.1.2 C-spec—Reset App_Comm_State STD

a. *Input/Output Flows*

Flow Name	*Type*
Network_Asleep	INPUT
App_Rx_Enabled	OUTPUT
App_Tx_Enabled	OUTPUT

b. *Operation*—The following shall be performed continually while the process which contains this C-spec is active:

This process shall behave functionally equivalent to the State Transition Diagram shown in Figure 35.

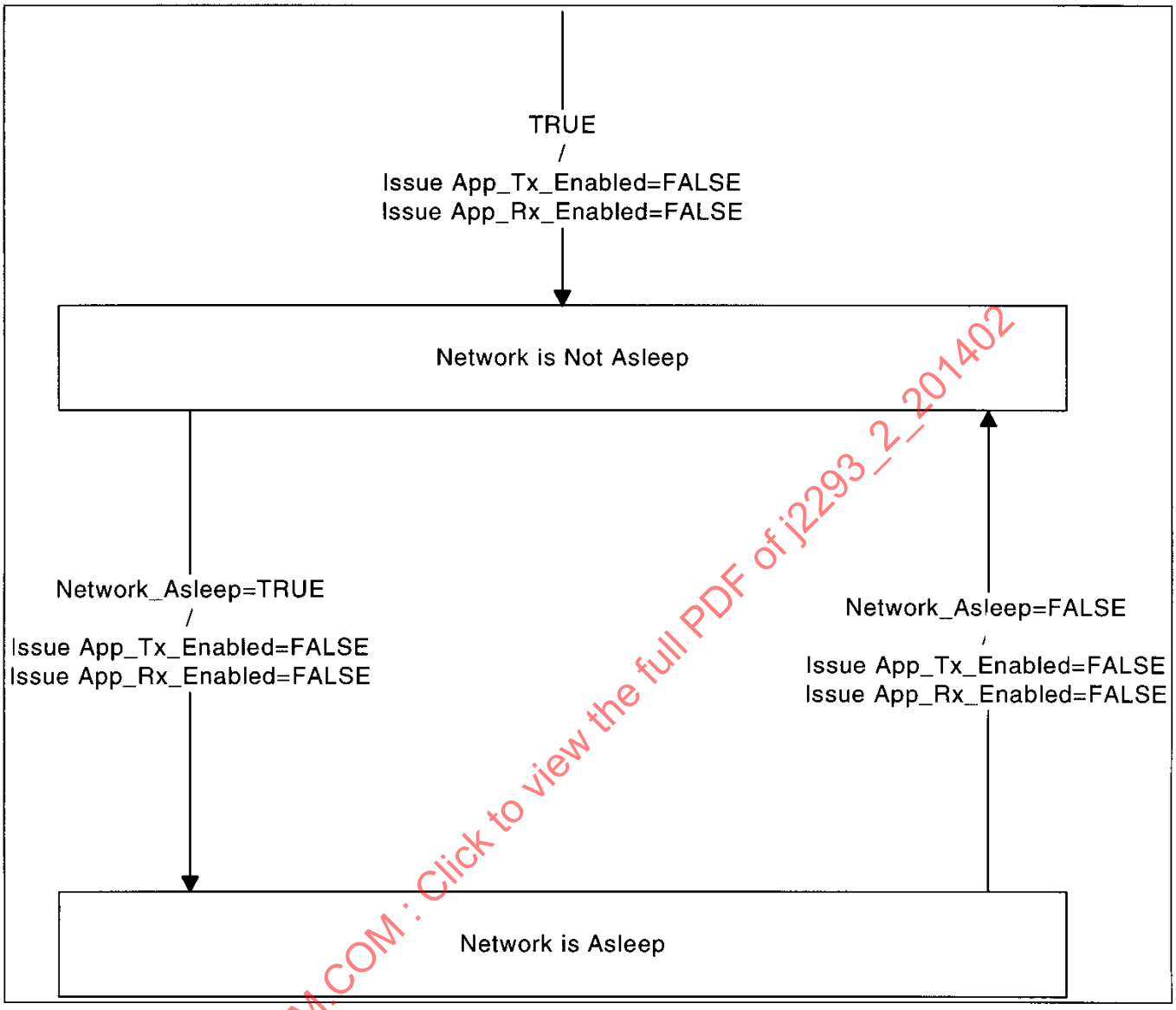

FIGURE 35 - RESET ETS_APP_COMM_STATE STD

6.5.2.4.1.3 C-spec—Enable App Service Requests PAT

a. *Input/Output Flows*

1. *Flow NameType*
App_Service_Request_Enable INPUT

2. *Process Controlled*—6.5.2.4.4

b. *Initialization*—The following shall be performed once upon activation of this process:

All processes controlled by this C-spec shall be disabled.

c. *Operation*—The following shall be performed continually while the process which contains this C-spec is active:

The process 6.5.2.4.4 shall be enabled and disabled based on the state of the control flow App_Service_Request as shown in Table 30.

TABLE 30 - LAUNCH APP SERVICE REQUEST PAT

Control Flow App_Service_Request_Enable	Process: 6.5.2.4.4 P-spec: Launch App Service Request
FALSE	DISABLED
TRUE	ENABLED

6.5.2.4.2 P-spec—Accept this App's Control Messages

If the message in Valid_Rcvd_Message is the "App_ Comm_State" or "App_Service_Request_Enable" message, this process decodes the "App Type" and "App ID" field in the message to see if it is directed towards this application. If the message is directed towards this application, then it is accepted and passed to other processes for further decoding.

If the message in Valid_Rcvd_Message is the "App_Comm_State" message, and the "App Type" and "App ID" fields are set to "ALL" applications and the fields Tx_Enabled and Rx_Enabled are set to FALSE then the message is accepted and passed to other processes for further decoding. The other cases of Tx_Enabled and Rx_Enabled values are rejected to prevent inadvertent enabling of all potential applications on the network.

If the message in Valid_Rcvd_Message is the "App_Service_Request_Enable" message, and the "App Type" and "App ID" fields in the message are set to "ALL" applications and the field App_Service_Request_Enable is set to FALSE then the message is accepted and passed to other processes for further decoding. The other case of App_Service_Request_Enable is rejected to prevent inadvertent enabling of all potential applications on the network.

a. *Input/Output Flows*

Flow Name	*Type*
Valid_Rcvd_Message	INPUT
This_App_Type	INPUT
This_App_ID	INPUT
Rcvd_App_Control_Message	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—The following shall be performed continually while this process is active:

1. If the parameter name of Valid_Rcvd_Message is "App_Comm_State" or "App_Service_Request_ Enable" and the operator is "LD" then

   If the App_Type = This_App_Type and App_ID = This_App_ID then
   Issue Rcvd_App_Control_Message with the value of Valid_Rcvd_Message.

2. If the parameter name of Valid_Rcvd_Message is "App Comm State" and operator is "LD" then

   If the App_Type=ALL and App_ID=ALL and Tx_Enabled=FALSE and Rx_Enabled=FALSE then
   Issue Rcvd_App_Control_Message with the value of Valid_Rcvd_Message.

3. If the parameter name of Valid_Rcvd_Message is "App_Service_Request_Enable" and operator is "LD" then

   If the App_Type=ALL and App_ID=ALL and App_Service_Request_Enable=FALSE then
   Issue Rcvd_App_Control_Message with the value of Valid_Rcvd_Message.

6.5.2.4.3 P-spec—Maintain App State

This process receives messages for App_Rx_Enabled and App_Tx_ Enabled and maintains their value based on the messages received and the history of events on the network. This message is expected to be sent using an on change strategy. The 6.9 is configured to default the message based on expected errors in an on change strategy or in the absence of messages.

a. *Input/Output Flows*

Flow Name	*Type*
Rcvd_App_Control_Message	INPUT
App_Rx_Enabled	OUTPUT
App_Tx_Enabled	OUTPUT
App_Control_Message	OUTPUT

b. *Operation*—This process shall perform functionally identical to 6.9 with the substitutions and exceptions which follow.

1. The context flows for 6.9 shall be substituted as defined in Table 31.

TABLE 31 - MAINTAIN ETS APP STATE CONTEXT SUBSTITUTIONS

Context Flow	Local Substitution Flow
Mode	—
Used_Message	—
Transmitter_Fault	—
Rx_Value	App_Rx_Enabled + App_Tx_Enabled
Launched_Message	App_Control_Message
Valid_Rcvd_Message	Rcvd_App_Control_Message

2. The configuration stores in 6.9 shall be substituted with the values defined in Table 48 for the Receive On Change Strategy.

3. Parameter_Name shall be substituted with "App Comm State".

4. Default_Value shall be assigned App_Tx_Is_Enabled = FALSE, App_Rx_Is_Enabled = FALSE.

6.5.2.4.4 P-spec—Launch App Service Request

1. *Input/Output Flows*

Flow Name	*Type*
App_Control_Message	INPUT
App_Service_Request	OUTPUT
Rcvd_App_Control_Message	OUTPUT

b. *Operation*—This process shall perform functionally identical to 6.8 with the substitutions and exceptions which follow.

1. The context flows for 6.8 shall be substituted as defined in Table 32.

TABLE 32 - LAUNCH APP SERVICE REQUEST CONTEXT SUBSTITUTIONS

Context Flow	Local Substitution Flow
Mode	—
Trigger_Launch	—
Tx_Value	TRUE
Launched_Message	App_Control_Message
Valid_Rcvd_Message	Rcvd_App_Control_Message

2. The configuration stores in 6.8 shall be have the values specified in Table 44 for Launch on Change Strategy assigned to them.

3. Parameter_Name shall be assigned "App_Service_Request".

6.5.2.4.5 P-spec—Maintain App Service Request Enable

a. *Input/Output Flows*

Flow Name	*Type*
Rcvd_App_Control_Message	INPUT
App_Service_Request_Enable	OUTPUT
App_Control_Message	OUTPUT

b. *Operation*—This process shall perform functionally identical to 6.9 with the substitutions and exceptions which follow.

1. The context flows for 6.9 shall be substituted as defined in Table 33.

TABLE 33 - MAINTAIN APP SERVICE REQUEST ENABLE CONTEXT SUBSTITUTIONS

Context Flow	Local Substitution Flow
Mode	—
Used_Message	—
Transmitter_Fault	—
Rx_Value	App_Service_Request_Enable
Launched_Message	App_Control_Message
Valid_Rcvd_Message	Rcvd_App_Control_Message

2. The configuration stores in 6.9 shall be substituted with the values defined in Table 48 for the Receive On Change Strategy.

3. Parameter_Name shall be substituted with "App_Service_Request_Enable".

4. Default_Value shall be assigned TRUE.

6.5.2.4.6 P-spec—Assign this App ID's Data to Message

a. *Input/Output Flows*

Flow Name	*Type*
App_Control_Message	INPUT
This_App_ID	INPUT
This_App_Type	INPUT
Launched_Message	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process.

None.

c. *Operation*—While this process is active, the following shall be performed upon receipt of App_Control_Message:

Issue Launched_Message with the App ID Data Field and the App Type data fields assigned the value of This_App_ID and This_App_Type per the encoding defined in SAE J2178.

6.5.2.4.7 Storage Requirements

The parent process contain storage for the following dataflows. These storage value shall maintain the last value received while the parent process is continuously enabled.

a. App_Tx_Enabled

b. App_Rx_Enabled

c. App_Service_Request_Enabled

d. This_App_ID

e. This_App_Type

6.5.2.5 P-spec—Indicate Application is Active in Node

a. *Input/Output Flows*

Flow Name	*Type*
EVSE_Launched_Message	OUTPUT

b. *Operation*—This process shall perform functionally identical to 6.8 with the substitutions and exceptions which follow.

1. The context flows for 6.8 shall be substituted as defined in Table 34.

TABLE 34 - INDICATE APPLICATION IS ACTIVE CONTEXT FLOW SUBSTITUTIONS

Context Flow	Local Substitution Flow
Mode	—
Trigger_Launch	—
Tx_Value	—
Launched_Message	Launched_Message
Valid_Rcvd_Message	—

2. The configuration stores in 6.8 shall be have the values specified in Table 44 for the Launch Periodic Strategy assigned to them.

3. Parameter_Name shall be assigned "App_Service_Request".

4. Launch_Period shall be assigned the value to 2 seconds.

6.5.2.6 P-spec—Configure Communication

a. *Input/Output Flows*

Flow Name	*Type*
Coupling_Type	INPUT
Bus_State	INPUT
Network_Asleep	INPUT
Bit_Encoding	OUTPUT
Data_Rate	OUTPUT
IFR_Enabled	OUTPUT
Bus_Media	OUTPUT

b. *Initialization*—The following shall be performed upon activation of this process.

Bit_Encoding, Data_Rate, Bus_Media and IFR_Enabled shall be assigned the value which the manufacturer designates as the most likely configuration of the Bus during operation.
The combinations of Bit_Encoding, Data_Rate, Bus_Media and IFR_Enabled shall only be assigned value which form valid implementations as defined in SAE J1850.

c. *Operation*

The following shall be performed once following the transition of Network_Asleep from TRUE to FALSE:

Using Bus_State and Coupling_Type, determine the value of

Bit_Encoding,
Data_Rate,
Bus_Media, and
IFR_Enabled
by the end of the first valid message.

6.5.2.7 P-spec—Detect Network Asleep

This process detects when the network is asleep as defined in SAE J2178/5.

a. *Input/Output Flows*

Flow Name	*Type*
Dual_Wire_Bus_State	INPUT
Single_Wire_Bus_State	INPUT
Bus_Media	INPUT
Network_Asleep	OUTPUT

b. *Initialization*—The following shall be performed upon activation of this process.

Network_Asleep=TRUE.

c. *Operation*—The following shall be performed continually while this process is active:

1. If Bus_Media is DUAL_WIRE and the value of Dual_Wire_Media is continuously PASSIVE for the preceding 10 seconds ± 1 second then

   assign Network_Asleep to TRUE
   else
   assign Network_Asleep to FALSE.

2. If Bus_Media is SINGLE_WIRE the value of Single_Wire_Media is continuously PASSIVE for the preceding 10 seconds ± 1 second then

   assign Network_Asleep to TRUE
   else
   assign Network_Asleep to FALSE.

6.5.2.8 Store—IFR_Enabled

For nodes which are located on the vehicle, this store is assigned a value by the OEM.
For nodes which reside off-board, this store contains the value of the last vehicle configuration the node communicated with.

6.5.2.9 Store—Bus_Type

For nodes which are located on the vehicle, this store is assigned a value by the OEM.
For nodes which reside off-board, this store contains the value of the last vehicle configuration the node communicated with.

6.5.3 Data Dictionary for Manage Application Slave Node Communication

Table 35 contains the definitions for the Launch Message Template data and control flows. All flows are described using Backus-Naur form (BNF). Any element in the Data Dictionary which has an unassigned value will be referred to as having a NULL value in the document.

TABLE 35 - MANAGE APPLICATION SLAVE NODE COMMUNICATION TEMPLATE DATA DICTIONARY

NAME	Contained in / Consists of	Units	Range	Resolution	Description
App_Control_Message	= (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	
App_Rx_Enabled	Context Flow = [TRUE | FALSE]	—	—	—	
App_Service_Request	Context Flow = [TRUE | FALSE]	—	—	—	
App_Tx_Enabled	Context Flow = [TRUE | FALSE]	—	—	—	
Bit_Encoding	In: Bus_Type = [VPW | PWM]	—	—	—	
Bus_State	Context_Flow = Dual_Wire_Bus_State + Single_Wire_Bus_State	—	—	—	
Bus_Type	= Bit_Encoding + Data_Rate + Media	—	—	—	
Byte_Data	In: Launched_Fbed_Message Launched_IDed_Message Launched_Message, Message_Queue Raw_Message, Tx_Message_w_Addr Tx_Message_w_CRC Tx_Message_w_Data Valid_CRC_Message, Valid_First_Byte_Message, Valid_Rcvd_Message, Tx_Buffer, Requeue_Message, Rcvd_App_Control_Message, App_Control_Message,	none	0 to 255	1	
Coupling_Type	Context Flow = [J1773 | J1772]				
Data_Rate	In: Bus_Type = [10.4K | 41.6K]	—	—	—	
Dual_Wire_Bus_State	In: Bus_State = [ACTIVE | PASSIVE]	—	—	—	

TABLE 35 - MANAGE APPLICATION SLAVE NODE COMMUNICATION TEMPLATE DATA DICTIONARY (CONTINUED)

NAME	Contained in / Consists of	Units	Range	Resolution	Description
Dual_Wire_Transmission	In: Transmission = [ACTIVE | PASSIVE]	—	—	—	
IFR_Enabled	= [TRUE | FALSE]	—	—	—	
Type_1_IFR_Required	= [TRUE | FALSE]	—	—	—	
Launched_FBed_Message	= (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	
Launched_IDed_Message	= (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	
Launched_Message	Context Flow = (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	
Location	Context Flow = [ONBOARD | OFF-BOARD]				
Bus_Media	In: Bus_Type = [DUAL_WIRE | SINGLE_WIRE]	—	—	—	
Network_Asleep	= [TRUE | FALSE]	—	—	—	
Network_Initialized	= [TRUE | FALSE]	—	—	—	
Operator	In: Valid_Rcvd_Message, Launched_Message, Launched_IDed_Message, Launched_Fbed_Message, Message_Queue, Tx_Message_w_Data, Tx_Message_w_Addr, Tx_Message_w_CRC, Tx_Buffer, Requeue_Message, App_Control_Message, Rcvd_App_Control_Message, = [LD | RPT | RQCV | REQ]	—	—	—	

TABLE 35 - MANAGE APPLICATION SLAVE NODE COMMUNICATION TEMPLATE DATA DICTIONARY (CONTINUED)

NAME	Contained in / Consists of	Units	Range	Resolution	Description
Parameter_Name	In: Valid_Rcvd_Message, Launched_Message, Launched_IDed_Message, Launched_Fbed_Message, Message_Queue, Tx_Message_w_Data, Tx_Message_w_Addr, Tx_Message_w_CRC, Tx_Buffer, Requeue_Message, Rcvd_App_Control_Message, App_Control_Message, = [Name of any valid SAE J2178 Message]	—	—	—	
Primary_ID	In: Valid_Primary_IDs	none	0 to 254	2	

Queue	= 0 {
	(Parameter_Name) + (Operator) +
	(0 { Byte_Data } 12)
	}
		—	—	—	

Raw_Message	= (0 { Byte_Data } 12)	—	—	—	
Rcvd_App_Control_Message	= (Parameter Name) + (Operator) +(0 { byte data } 12)	—	—	—	
Requeue_Message	= (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	
Single_Wire_Bus_State	In: Bus_State = [ACTIVE | PASSIVE]	—	—	—	
Single_Wire_Transmission	In: Transmission = [ACTIVE | PASSIVE]	—	—	—	
Source_Address		none	0 to 255	1	
This_App_ID	= [any of the values defined for App_ID in SAE J1278]	—	—	—	
This_App_Type	= [any of the values defined for App_Type defined in SAE J2178]	—	—	—	
Transmission	Context Flow = Dual_Wire_Transmission Single_Wire_Transmission	—	—	—	

TABLE 35 - MANAGE APPLICATION SLAVE NODE COMMUNICATION TEMPLATE DATA DICTIONARY (CONTINUED)

NAME	Contained in / Consists of	Units	Range	Resolution	Description
Tx_Buffer	= (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	
Tx_Data	Context_Flow = [the bundle of data which is communicated by the messages]	—	—	—	Tx_Data is an abstract description of the data flow which are communicated by the Application Slave Node. Since for most messages, data is required to be bound to the message immediately prior to transmission, this data is required at this level in the system.
Tx_Message_w_Addr	= (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	
Tx_Message_w_CRC	= (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	
Tx_Message_w_Data	= (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	
Valid_CRC_Message	= (0 { Byte_Data } 12)	—	—	—	
Valid_First_Byte_Message	= (0 { Byte_Data } 12)	—	—	—	
Valid_Primary_IDs	Context Flow = (0 { Primary_ID })				
Valid_Rcvd_Message	Context Flow = (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	
Wakeup_Required	= [TRUE | FALSE]	—	—	—	

6.6 Manage Application Master Node Communication Template

6.6.1 Context for Manage Application Master Node Communication Template

This template is applied throughout this document by first substituting flows from the local DFD in place of context flows shown in Figure 36.

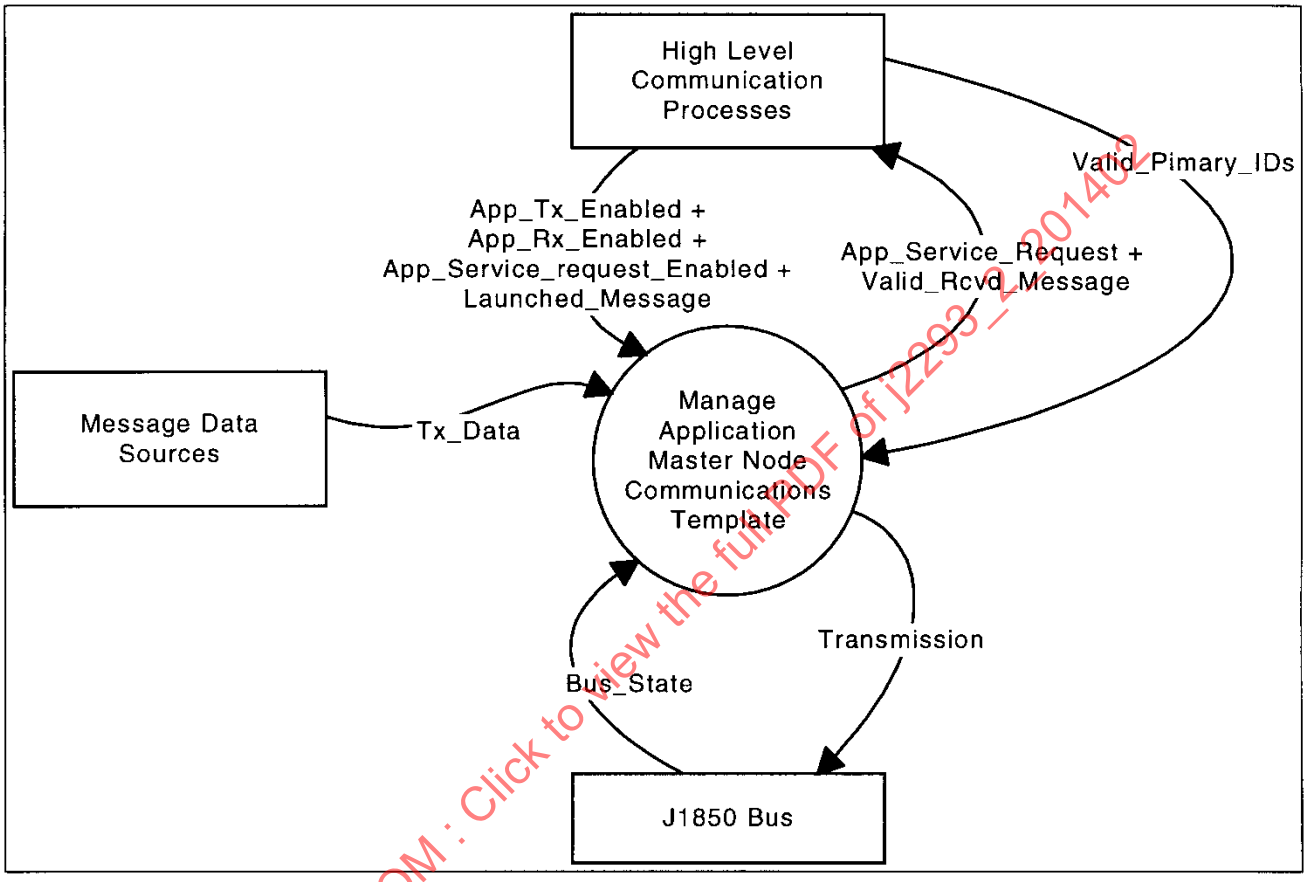

FIGURE 36 - MANAGE APPLICATION MASTER NODE COMMUNICATION CONTEXT DIAGRAM

6.6.2 Functional Requirements For Manage Application Master Node Communication Template

The Application Master Node Communication Template provides a set of requirements which define how a slave node

a. Receives messages from the J1850 bus

b. Detects which messages are directed to a specific part of an application

c. Decodes those messages to obtain information for the application dataflows

d. Determines when to update information in other parts of the application

e. Encodes application information in this node to send to other parts of the application in different nodes

f. Send information to other parts of an application using messages on the SAE J1850 Bus

g. Support basic fault recovery protocols

h. Provide controls application slave communication

Figure 37 illustrates the relationship between these processes.

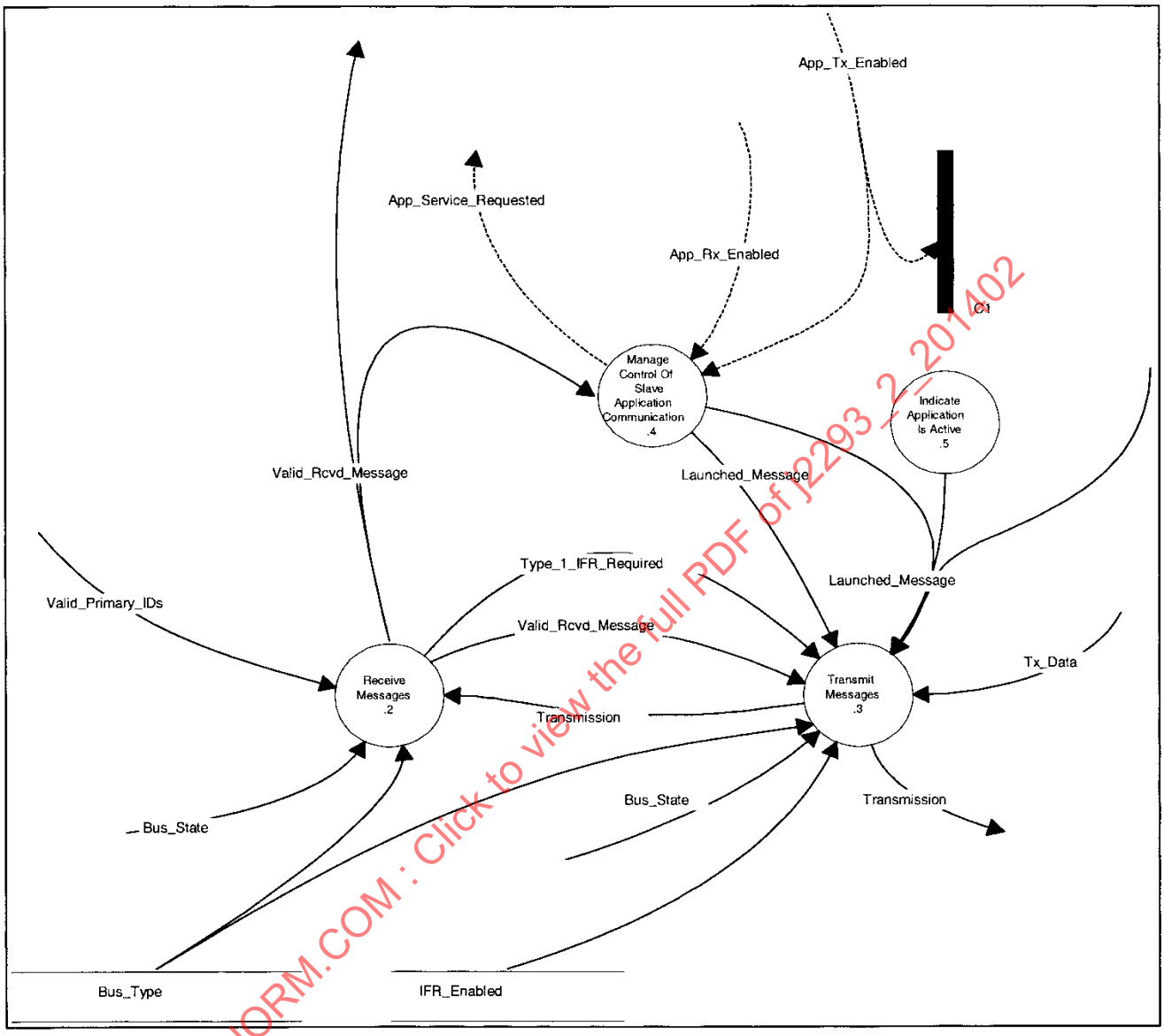

FIGURE 37 - MANAGE APPLICATION MASTER NODE COMMUNICATION DFD

6.6.2.1 Process Control Requirements

6.6.2.1.1 C-spec—Indicate Application is Active in Node PAT.

a. *Inputs, Outputs, Processes Controlled*

1. *Flow Name* *Type*
   App_Tx_Is_Enabled INPUT

2. Process Controlled—Indicate Application is Active in Node

b. *Initialization*—None.

c. *Operation*—The following shall be performed continually while the process which contains this C-spec is active:

The process identified in Table 24 shall be enabled and disabled based on the state of the control flow in Table 36.

TABLE 36 - INDICATE APP IS ACTIVE IN NODE PAT

Control Flow App_Tx_Enable	Process 6.5.2.5 P-spec—Indicate Application is Active in Node
FALSE	DISABLED
TRUE	ENABLED

6.6.2.2 Receive Messages

a. *Input/Output Flows*

Flow Name	*Type*
Bus_State	INPUT
Bus_Type	INPUT
Transmission	INPUT
Valid_Primary_IDs	INPUT
Type_1_IFR_Required	OUTPUT
Valid_Rcvd_Message	OUTPUT

b. *Operation*—This process shall perform functionally equivalent to 6.5.2.2

6.6.2.3 Transmit Messages

a. *Input/Output Flows*

Flow Name	*Type*
Bus_State	INPUT
Bus_Type	INPUT
Type_1_IFR_Required	INPUT
Launched_Message	INPUT
Tx_Data	INPUT
Valid_Rcvd_Message	INPUT
Transmission	OUTPUT

b. *Operation*—This process shall perform functionally equivalent to 6.5.2.2.

6.6.2.4 Manage Control of Slave Application Communication

Figure 33 provides an illustration of the relationship between these processes. See Figure 38.

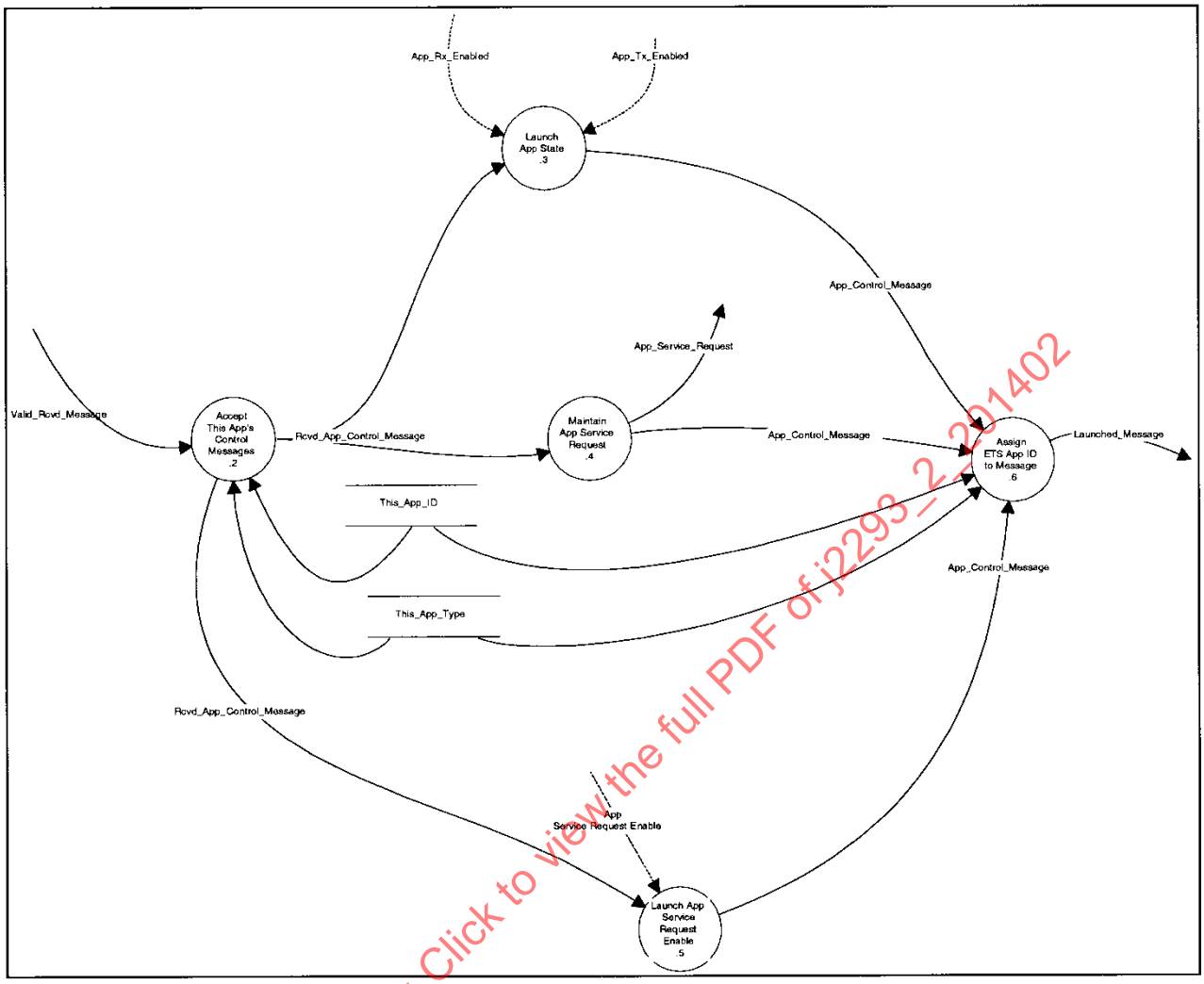

FIGURE 38 - MANAGE CONTROL OF APPLICATION SLAVE COMMUNICATION DFD

6.6.2.4.1 Process Control Requirements

6.6.2.4.2 P-spec: Accept App Control Messages

If the message in Valid_Rcvd_Message is the "App_Service_ Request" message, this process decodes the "App Type" and "App ID" field in the message to see if it is directed towards this application. If the message is directed towards this application, then it is accepted and passed to other processes for further decoding.

a. *Input/Output Flows*

Flow Name	*Type*
Valid_Rcvd_Message	INPUT
This_App_Type	INPUT
This_App_ID	INPUT
Rcvd_App_Control_Message	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—The following shall be performed continually while this process is active:

1. If the parameter name of Valid_Rcvd_Message is "App_Service_Request" and the operator is "LD" then

   If the App_Type = This_App_Type and App_ID = This_App_ID then
   Issue Rcvd_App_Control_Message with the value of Valid_Rcvd_Message.

6.6.2.4.3 P-spec—Launch App Service Request Enable

a. *Input/Output Flows*

Flow Name	*Type*
Rcvd_App_Control_Message	INPUT
App_Service_Request_Enable	OUTPUT
App_Control_Message	OUTPUT

b. *Operation*—This process shall perform functionally identical to 6.8 with the substitutions and exceptions which follow.

1. The context flows for 6.8 shall be substituted as defined in Table 37.

TABLE 37 - LAUNCH APP STATE CONTEXT SUBSTITUTIONS

Context Flow	Local Substitution Flow
Mode	—
Trigger_Launch	—
Tx_Value	App_Service_Requeue_Enable
Launched_Message	App_Control_Message
Valid_Rcvd_Message	Rcvd_App_Control_Message

2. The configuration stores in 6.8 shall be substituted with the values specified in Table 44 for a Launch On Change Strategy. Parameter Name shall be substituted with "App_Service_Request Enable".

6.6.2.4.4 P-spec—Maintain App Service Request

a. *Input/Output Flows*

Flow Name	*Type*
App_Control_Message	INPUT
App_Service_Request_Enable	OUTPUT
Rcvd_App_Control_Message	OUTPUT

b. *Operation*—This process shall perform functionally identical to 6.9 with the substitutions and exceptions which follow.

1. The context flows for 6.9 shall be substituted as defined in Table 38.

TABLE 38 - MAINTAIN APP SERVICE REQUEST CONTEXT SUBSTITUTIONS

Context Flow	Local Substitution Flow
Mode	—
Used_Message	—
Transmitter_Fault	—
Rx_Value	App_Service_Request
Launched_Message	App_Control_Message
Valid_Rcvd_Message	Rcvd_App_Control_Message

2. The configuration stores in 6.9 shall be substituted as defined in Table 38 for a Launch On Change Strategy. Parameter_Name shall be substituted with "App_Service_Request".

6.6.2.4.5 P-spec—Launch App State

a. *Input/Output Flows*

Flow Name	*Type*
Rcvd_App_Control_Message	INPUT
App_Rx_Enabled	INPUT
App_Tx_Enabled	INPUT
App_Control_Message	OUTPUT

b. *Operation*—This process shall perform functionally identical to 6.8 with the substitutions and exceptions which follow.

1. The context flows for 6.8 shall be substituted as defined in Table 39.

TABLE 39 - LAUNCH APP STATE CONTEXT SUBSTITUTIONS

Context Flow	Local Substitution Flow
Mode	—
Trigger_Launch	—
Tx_Value	App_Rx_Enabled + App_Tx_Enabled
Launched_Message	App_Control_Message
Valid_Rcvd_Message	Rcvd_App_Control_Message

2. The configuration stores in 6.8 shall be substituted with the values specified in Table 44 for a Launch On Change Strategy. Parameter Name shall be substituted with "App_Comm_State".

6.6.2.4.6 P-spec—Create App ID Data in Message

a. *Input/Output Flows*

Flow Name	*Type*
App_Control_Message	INPUT
This_App_ID	INPUT
This_App_Type	INPUT
Launched_Message	OUTPUT

b. *Operation*—This process shall perform functionally identical to 6.5.2.4.6

6.6.2.4.7 Storage Requirements

The parent process contain storage for the following dataflows. These storage values are read only storage. The value for each of these stores are assigned when this templates use is specified.

a. This_App_ID

b. This_App_Type

6.6.2.5 P-spec: Indicate Application is Active in Node

a. *Input/Output Flows*

Flow Name	*Type*
EVSE_Launched_Message	OUTPUT

b. *Operation*—This process shall perform functionally identical to 6.5.2.5

6.6.2.6 Storage Requirements

The parent process contain storage for the following dataflows. These storage values are read only storage. The value for each of these stores are assigned when this templates use is specified.

a. IFR_Enabled

b. Bus_Type

6.6.2.7 Data Dictionary for Manage Application Master Node Communication

Table 40 contains the definitions for the data and control flows in this template. All flows are described using Backus-Naur form (BNF). Any element in the Data Dictionary which has an unassigned value will be referred to as having a NULL value in the document.

TABLE 40 - MANAGE APPLICATION MASTER NODE COMMUNICATION TEMPLATE DATA DICTIONARY

NAME	Contained in / Consists of	Units	Range	Resolution	Description
App_Control_Message	= (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	
App_Rx_Enabled	Context Flow = [TRUE | FALSE]	—	—	—	
App_Service_Request	Context Flow = [TRUE | FALSE]	—	—	—	
App_Service_Request_Enabled	Context Flow = [TRUE | FALSE]	—	—	—	
App_Tx_Enabled	Context Flow = [TRUE | FALSE]	—	—	—	
Bit_Encoding	In: Bus_Type = [VPW | PWM]	—	—	—	
Bus_Media	In: Bus_Type = [DUAL_WIRE | SINGLE_WIRE]	—	—	—	
Bus_State	Context_Flow = Dual_Wire_Bus_State + Single_Wire_Bus_State	—	—	—	
Bus_Type	= Bit_Encoding + Data_Rate + Bus_Media	—	—	—	
Byte_Data	In: Launched_Message, Valid_Rcvd_Message, Rcvd_App_Control_Message, App_Control_Message;	none	0 to 255	1	
Data_Rate	In: Bus_Type = [10.4K | 41.6K]	—	—	—	
Dual_Wire_Bus_State	In: Bus_State = [ACTIVE | PASSIVE]	—	—	—	
Dual_Wire_Transmission	In: Transmission = [ACTIVE | PASSIVE]	—	—	—	
IFR_Enabled	= [TRUE | FALSE]	—	—	—	
Launched_Message	Context Flow = (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	

TABLE 40 - MANAGE APPLICATION MASTER NODE COMMUNICATION TEMPLATE DATA DICTIONARY (CONTINUED)

NAME	Contained in / Consists of	Units	Range	Resolution	Description
Operator	In: Valid_Rcvd_Message, Launched_Message, App_Control_Message, Rcvd_App_Control_Message, = [LD | RPT | RQCV | REQ]	—	—	—	
Parameter_Name	In: Valid_Rcvd_Message, Launched_Message, Rcvd_App_Control_Message, App_Control_Message, = [Name of any valid SAE J2178 Message]	—	—	—	
Primary_ID	In: Valid_Primary_IDs	none	0 to 254	2	
Rcvd_App_Control_Message	= (Parameter Name) + (Operator) +(0 { byte data } 12)	—	—	—	
Single_Wire_Bus_State	In: Bus_State = [ACTIVE | PASSIVE]	—	—	—	
Single_Wire_Transmission	In: Transmission = [ACTIVE | PASSIVE]	—	—	—	
This_App_ID	= [any of the values defined for App_ID in SAE J1278]	—	—	—	
This_App_Type	= [any of the values defined for App_Type defined in SAE J2178]	—	—	—	
Transmission	Context Flow = Dual_Wire_Transmission Single_Wire_Transmission	—	—	—	
Tx_Data	Context_Flow = [the bundle of data which is communicated by the messages]	—	—	—	Tx_Data is an abstract description of the data flow which are communicated by the Application Slave Node. Since for most messages, data is required to be bound to the message immediately prior to transmission, this data is required at this level in the system.
Type_1_IFR_Required	= [TRUE | FALSE]	—	—	—	
Valid_Primary_IDs	Context Flow = (0 { Primary_ID })				
Valid_Rcvd_Message	Context Flow = (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	

6.7 Detect Application Transmitter Fault Template

6.7.1 Context for Detect Application Transmitter Fault Template

This template is applied throughout this document by first substituting flows from the local DFD in place of context flows shown in Figure 39.

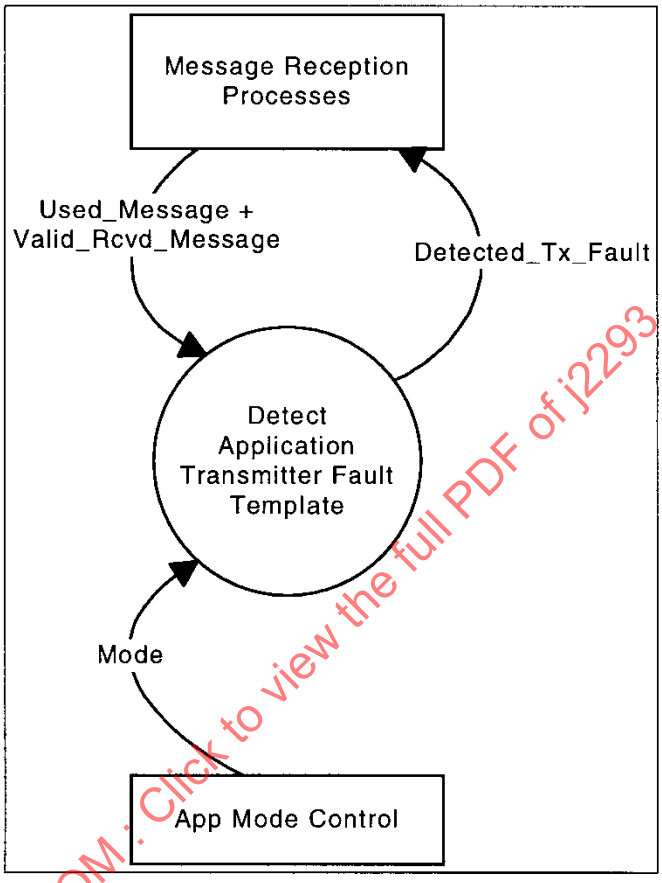

FIGURE 39 - DETECT APPLICATION TRANSMITTER FAULT TEMPLATE CONTEXT DIAGRAM

6.7.2 Functional Requirements for Detect Application Transmitter Fault Template

This template de- scribes the processes which interact to detect faults with transmitters used by an application. These processes learn the transmitters which are in use by the application and detect faults when any of those transmitter stop transmitting. These processes reset the learning any time there is an application mode change. The specific number of processes which are required to be monitored by this template are specified when it is applied.

Figure 40 illustrates the relationship between the process which perform these functions.

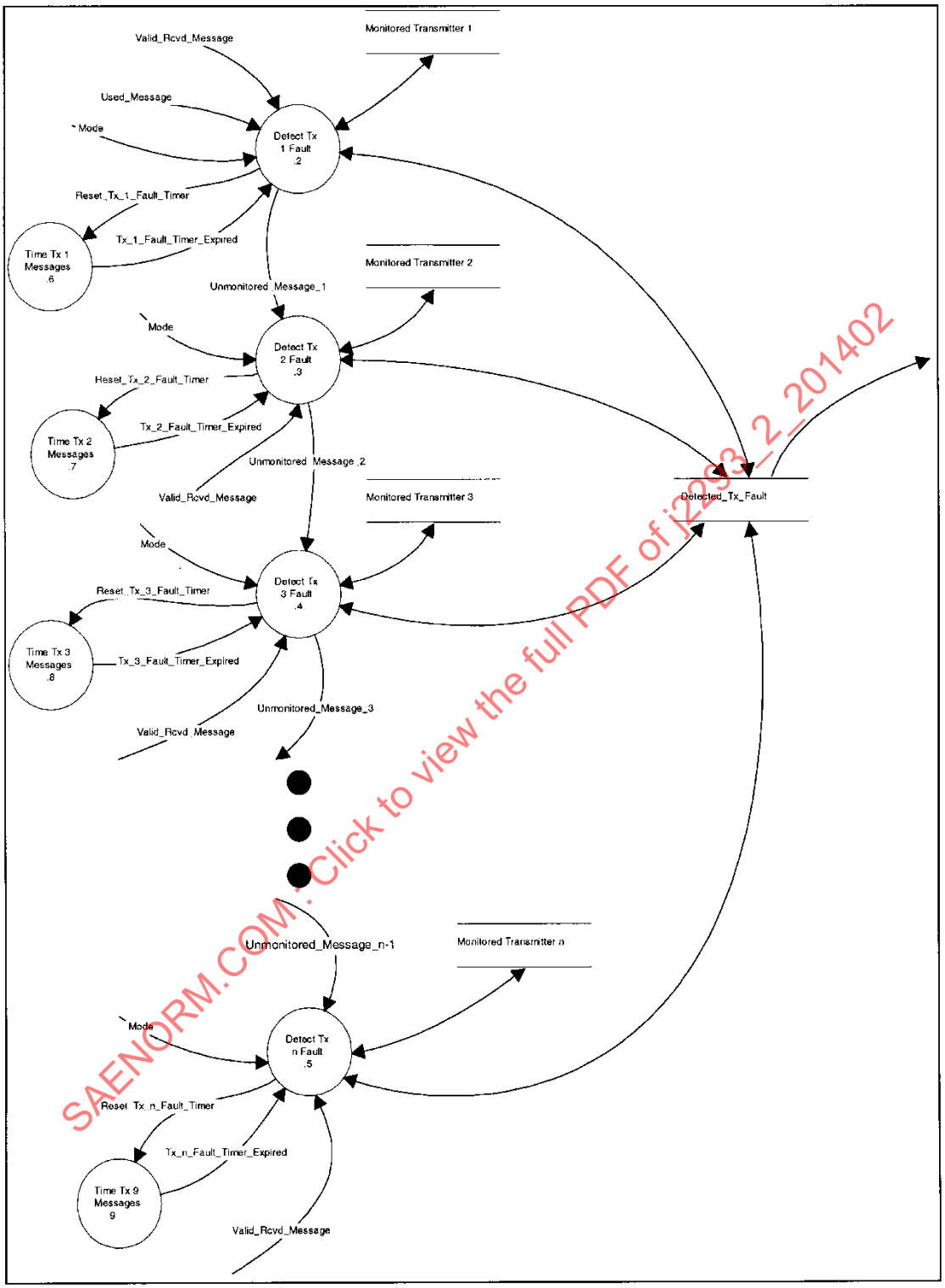

FIGURE 40 - DETECT APPLICATION TRANSMITTER FAULT DFD

6.7.2.1 Process Control Requirements

None.

6.7.2.2 P-spec—Detect Tx_1 Fault

This process learns a single transmitter's source address and detects faults in that transmitter. It only learns transmitter addresses for transmitters which source messages which are used by the application. After learning the transmitter's address, this process monitors the system message traffic and detects when that transmitter is no longer actively transmitting. Should this occur, the transmitter's address is added to a list of faulted transmitters. This is used to communicate with the processes which receive messages to trigger their defaulting and message recovery strategies.

Figure 41 provides an overview of the operation of this process.

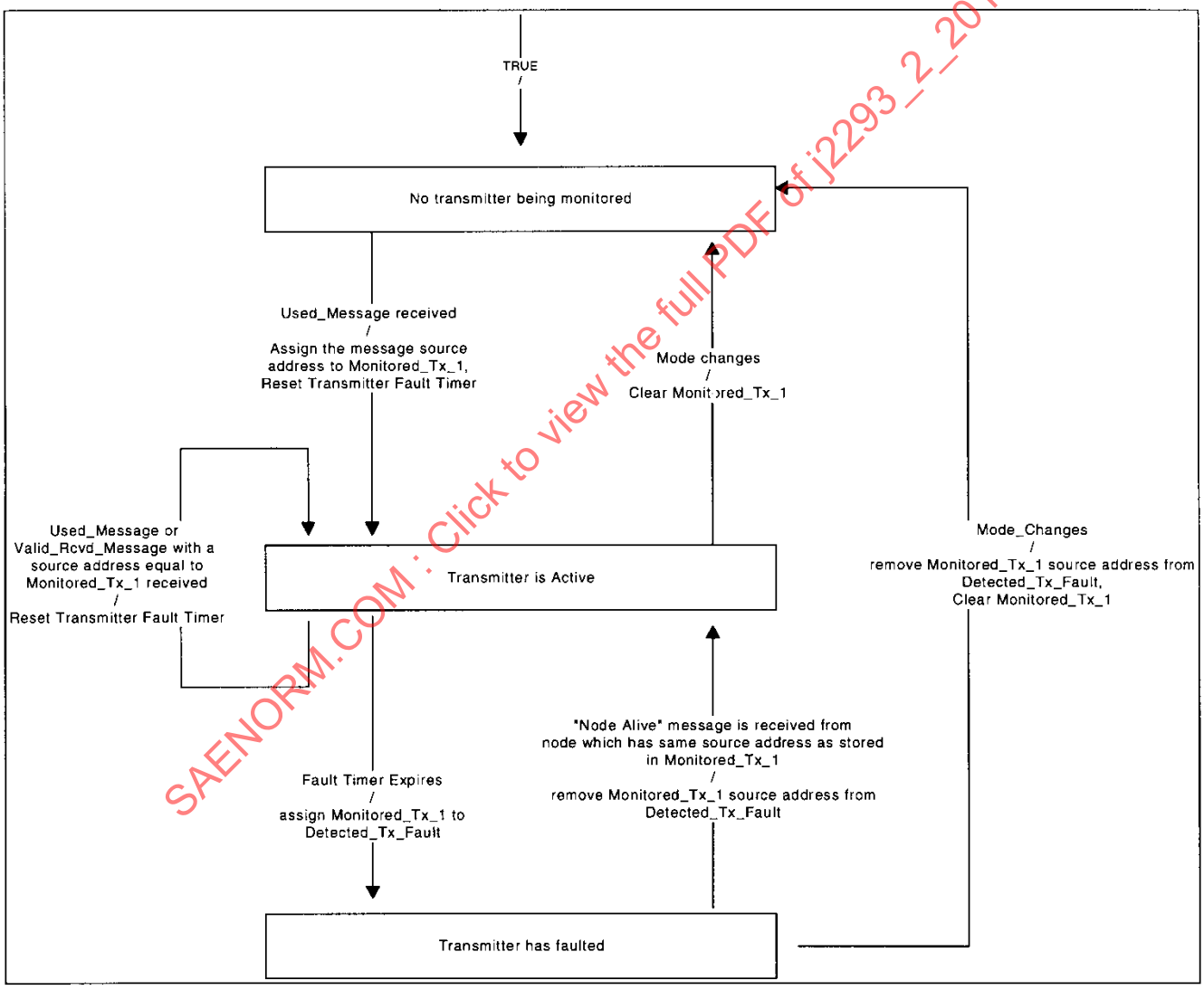

FIGURE 41 - ILLUSTRATION OF DETECT TX FAULT STD

a. *Input/Output Flows*

Flow Name	*Type*
Valid_Rcvd_Message	INPUT
Used_Message	INPUT
Mode	INPUT
Monitored_Transmitter_1	INPUT/OUTPUT
Tx_1_Fault_Timer_Expired	INPUT
Reset_Tx_1_Fault_Timer	OUTPUT
Unmonitored_Message_1	OUTPUT
Detected_Tx_Fault	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process.

Monitored_Transmitter_1 shall be cleared.

c. *Operation*

1. While this process is active, the following shall be performed on receipt of Used_Message:

   If Monitored_Transmitter_1 does not have a message source address stored in it, then
   assign the address from Used_Message to Monitored_Transmitter_1,
   Issue Reset_Transmitter_1_Fault_Timer.

   If Monitored_Transmitter_1 has a message source address stored in it then
   If the source address in Used_Message is equal to the address in Monitored_Transmitter_1 then
   Issue Reset_Transmitter_1_Fault_Timer.
   else
   Assign Unmonitored_Message_1 the value of Used_Message.
   Issue Unmonitored_Message_1.

2. While this process is active, the following shall be performed on receipt of Valid_Rcvd_Message:

   If Monitored_Transmitter_1 has a message source address stored in it and the source address in Valid_Rcvd_Message is equal to the address in Monitored_Transmitter_1 then
   Issue Reset_Transmitter_1_Fault_Timer.

3. While this process is active, the following shall be performed on any change in Mode:

   Clear Monitored_Transmitter_1.

4. While this process is active, the following shall be performed on receipt of Tx_1_Fault_Timer_Expired=TRUE:

   If Monitored_Transmitter_1 contains a valid message source address then

   Assign value of Monitored_Transmitter_1 to Detected_Tx_Fault without affecting any values which are currently in this list of faults.

5. While this process is active, the following shall be performed on receipt of Valid_Rcvd_Message with a parameter name of "Node Alive" and a source address equal to Monitored_Transmitter:

   If the source address stored in Monitored_Transmitter is in the Detected_Tx_Fault list, then remove it from that list.

6.7.2.3 P-spec—Detect Tx_2 Fault

This process is functionally equivalent to 6.7.2.2 with the following changes:

a. Substitute Unmonitored_Message_1 for Used_Message.

b. Substitute Monitored_Transmitter_2 for Monitored_Transmitter_1.

c. Substitute Tx_2_Fault_Timer_Expired for Tx_1_Fault_Timer_Expired.

d. Substitute Reset_Tx_2_Fault_Timer for Reset_Tx_1_Fault_Timer.

e. Substitute Unmonitored_Message_2 for Unmonitored_Message_1.

6.7.2.4 P-spec—Detect Tx_3 Fault

This process is functionally equivalent to 6.7.2.2 with the following changes:

a. Substitute Unmonitored_Message_2 for Used_Message.

b. Substitute Monitored_Transmitter_3 for Monitored_Transmitter_1.

c. Substitute Tx_3_Fault_Timer_Expired for Tx_1_Fault_Timer_Expired.

d. Substitute Reset_Tx_3_Fault_Timer for Reset_Tx_1_Fault_Timer.

e. Substitute Unmonitored_Message_3 for Unmonitored_Message_1.

6.7.2.5 P-spec—Detect Tx_n Fault

This process is functionally equivalent to 6.7.2.2 with the following changes:

a. Substitute Unmonitored_Message_n-1 for Used_Message.

b. Substitute Monitored_Transmitter_n for Monitored_Transmitter_1.

c. Substitute Tx_n_Fault_Timer_Expired for Tx_1_Fault_Timer_Expired.

d. Substitute Reset_Tx_n_Fault_Timer for Reset_Tx_1_Fault_Timer.

e. Substitute Unmonitored_Message_n for Unmonitored_Message_1.

6.7.2.6 P-spec—Time Tx_1 Messages

a. Input/Output Flows

Flow Name	*Type*
Reset_Tx_1_Fault_Timer	INPUT
Tx_1_Fault_Timer_Expired	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process.

Assign Tx_1_Fault_Timer_Expired the value of FALSE.

c. *Operation*—The following shall be performed continually while this process is active:

ssign Tx_1_Fault_Timer_Expired the value of TRUE if Reset_Tx_1_Fault_Timer has not been TRUE continuously for the previous 5 seconds (±500 msec), otherwise assign Tx_1_Fault_Timer_Expired the value of FALSE.

6.7.2.7 P-spec—Time Tx_2 Messages

This process is functionally equivalent to 6.7.2.6 with the following changes:

a. Substitute Reset_Tx_2_Fault_Timer for Reset_Tx_1_Fault_Timer.

b. Substitute Tx_2_Fault_Timer_Expired for Tx_1_Fault_Timer_Expired.

6.7.2.8 P-spec—Time Tx_3 Messages

This process is functionally equivalent to 6.7.2.6 with the following changes:

a. Substitute Reset_Tx_3_Fault_Timer for Reset_Tx_1_Fault_Timer.

b. Substitute Tx_3_Fault_Timer_Expired for Tx_1_Fault_Timer_Expired.

6.7.2.9 P-spec—Time Tx_n Messages

This process is functionally equivalent to 6.7.2.6 with the following changes:

a. Substitute Reset_Tx_n_Fault_Timer for Reset_Tx_1_Fault_Timer.

b. Substitute Tx_n_Fault_Timer_Expired for Tx_1_Fault_Timer_Expired.

6.7.2.10 Storage Requirements

The parent process contain storage for the following dataflows. These storage value shall maintain the last value received while the parent process is continuously enabled.

a. Monitored_Transmitter_1

b. Monitored_Transmitter_2

c. Monitored_Transmitter_3

d. Monitored_Transmitter_n

e. Detected_Tx_Fault

Monitored_Transmitter_n is used to indicate that there is storage maintained for each instance of the data flow with the name "Monitored_Transmitter_" with a numerical suffix.

6.7.3 Data Dictionary for Manage Application Slave Node Communication

Table 41 contains the definitions for the data and control flows in this template. All flows are described using Backus-Naur form (BNF). Any element in the Data Dictionary which has an unassigned value will be referred to as having a NULL value in the document.

6.8 Launch Message Template

6.8.1 Context for Launch Message Template

This template provides a set of requirements for launching a message on the ETS network. These functional requirements encompass all of the launch strategies documented in SAE J2178/5 plus additional launch requirements required to support the ETS application. Most messages will only use a subset of these processes to schedule launches.

This template is applied throughout this document by first substituting flows from the local DFD in place of context flows shown in Figure 42. Second, the read only stores used throughout the template are assigned values to determine which launch strategies are applied to this message. Finally, a process activation table is used to control when this set of strategies will be active. This allows this template to describe any of the launch strategies supported.

TABLE 41 - DETECT APPLICATION TRANSMITTER FAULT TEMPLATE DATA DICTIONARY

NAME	Contained in / Consists of	Units	Range	Resolution	Description
Byte_Data	In: Used_Message, Valid_Rcvd_Message, Unmonitored_Message_1, Unmonitored_Message_2, Unmonitored_Message_3, Unmonitored_Message_n-1,	none	0 to 255	1	
Source_Address	In: Monitored_Transmitter_1, Monitored_Transmitter_2, Monitored_Transmitter_3, Monitored_Transmitter_n, Detected_Tx_Fault	none	0 to 255	1	
Monitored_Transmitter_1	= Source_Address	—	—	—	
Monitored_Transmitter_2	= Source_Address	—	—	—	
Monitored_Transmitter_3	= Source_Address	—	—	—	
Monitored_Transmitter_n	= Source_Address	—	—	—	
Reset_Tx_1_Timer	= [TRUE | FALSE]	—	—	—	
Reset_Tx_2_Timer	= [TRUE | FALSE]	—	—	—	
Reset_Tx_3_Timer	= [TRUE | FALSE]	—	—	—	
Reset_Tx_n_Timer	= [TRUE | FALSE]	—	—	—	
Tx_1_Fault_Timer_Expired	= [TRUE | FALSE]	—	—	—	
Tx_2_Fault_Timer_Expired	= [TRUE | FALSE]	—	—	—	
Tx_3_Fault_Timer_Expired	= [TRUE | FALSE]	—	—	—	
Tx_n_Fault_Timer_Expired	= [TRUE | FALSE]	—	—	—	
Mode	= [any of the modes the application may operate in]				
Detected_Tx_Fault	= (0 { Source_Address })				
Used_Message	Context Flow = (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	

TABLE 41 - DETECT APPLICATION TRANSMITTER FAULT TEMPLATE DATA DICTIONARY (CONTINUED)

NAME	Contained in / Consists of	Units	Range	Resolution	Description
Operator	In: Used_Message, Valid_Rcvd_Message, Unmonitored_Message_1, Unmonitored_Message_2, Unmonitored_Message_3, Unmonitored_Message_n-1, = [LD | RPT | RQCV | REQ]	—	—	—	
Parameter_Name	In: Used_Message, Valid_Rcvd_Message, Unmonitored_Message_1, Unmonitored_Message_2, Unmonitored_Message_3, Unmonitored_Message_n-1, = [Name of any valid SAE J2178 Message]	—	—	—	
Valid_Rcvd_Message	Context Flow = (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	
Unmonitored_Message_1	= (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	
Unmonitored_Message_2	= (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	
Unmonitored_Message_3	= (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	
Unmonitored_Message_n-1	= (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	

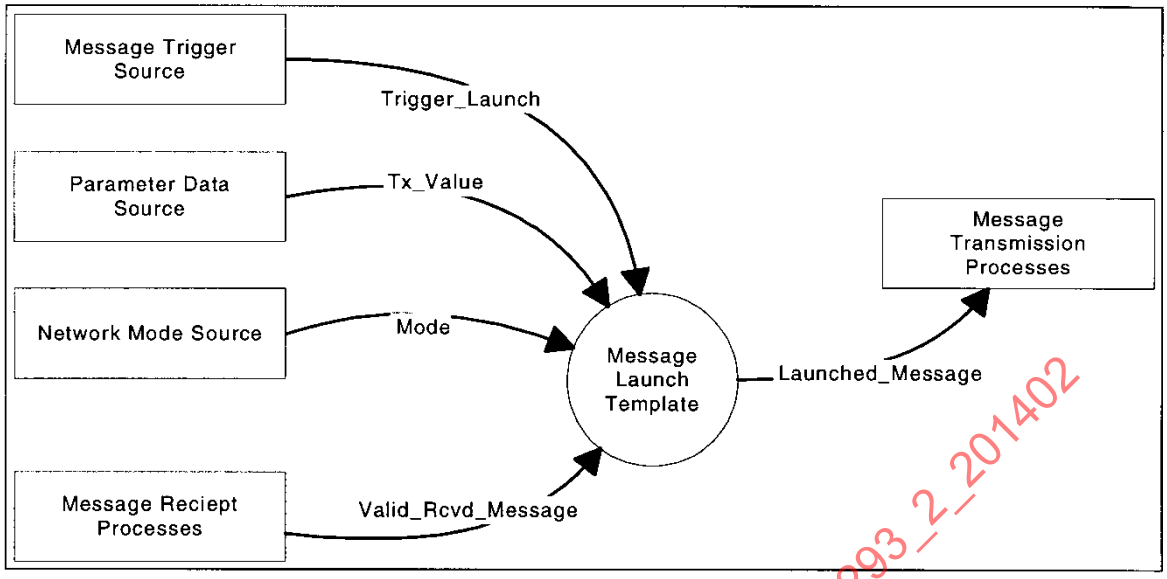

FIGURE 42 - LAUNCH MESSAGE TEMPLATE FUNCTIONAL CONTEXT

In addition to setting launch strategies, this template also configures part of a message's error recovery strategy. One part of this is having the transmitter insure that a transmission is received. This is accomplished by have the receiver acknowledge receipt of a message transmission. If the transmission is not received the transmitter relaunches the message until the message is received or a maximum number of retries is experienced. See Table 42.

This template is intended to only specify the functional requirements to launch a message. Therefore, the strategy of acknowledging that a message has been received by launching a message is detailed in the Maintain Parameter Template. See Tables 43 and 44.

TABLE 42 - SAE J2293-2 LAUNCH STRATEGIES

Launch Strategy Name	Launch Strategies
Launch Periodical	Launch a message to update parameter periodically
Launch On Change with Verification	Launch a message to update parameter on parameter value change, repeat launches until verification that it was received is detected
Launch On Change w/o Verification	Launch a message to update parameter on parameter value change, repeat launches until verification that it was received is detected
Launch On Init	Launch a message once on a mode change or initialization of a channel

TABLE 43 - LAUNCH MESSAGE TEMPLATE CONFIGURATION STORES

Store Name	Description
Launch_on_Activation	This is a flag which indicates that a message is launched on activation of the launch process.
Launch_on_Mode_Change	This is a flag which indicates that a message is launched on an application mode change.
Launch_on_Parameter_Change	This flag indicate that a message is launched on a parameter change which exceed the limits set in "Threshold_For_Parameter_Change"
Max_Number_of_Relaunches	This value is used to set the maximum number of relaunches of a message.
Max_Time_Between_Launches	This value set the maximum interval between launches of a message. This value is used to configure a message for periodic launches.
Message_Ack_Operator	This is the operator in a message which is sent by a receiving node to indicate that the message was received. A RPT operator is used in the ETS network to acknowledge a LD operator. This is the only acknowledgment operator used in the ETS system.
Message_Parameter_Name	This is the SAE J2178 parameter name associated with this message.
Message_Request_Operator	This is the operator sent in a message to indicate a request to send a parameter update. In the ETS network, a RQCV is used to request a LD update and a REQ is sent to request a RPT update.
Message_Update_Operator	This is the operator sent with a message to indicate that data is attached to the message and can be used to update a record of that parameter in a receiving node. In the ETS network, LD and RPT operators are used to indicate that a message contains data which can be used to update a record of the message's parameter.
Min_Time_Between_Launches	This value sets the minimum interval which must pass between launches of certain types of messages.
Threshold_For_Parameter_Change	This value set the minimum threshold for change in a parameter before the message is launched.
Time_Between_Relaunches	This value is used to set the maximum time between subsequent relaunches of a message due to lack of receiver response. If this interval has expired since either the first relaunch or last previous relaunch without an acknowledgment, the message is relaunched.
Time_to_First_Relaunch	This value is used to set the maximum interval between a message launch and an expected response from a receiver. If this interval expires and no response has been received, than the message is relaunched.

TABLE 44 - TYPICAL VALUES FOR LAUNCH STRATEGIES

Store Name	Launch Periodic	Launch On Change With Verification	Launch On Change w/o Verification	Launch on Init
Launch_on_Activation	TRUE	FALSE	FALSE	TRUE
Launch_on_Mode_Change	TRUE	TRUE	FALSE	TRUE
Launch_on_Parameter_Change	FALSE	TRUE	TRUE	FALSE
Max_Number_of_Relaunches	NOT USED	5	NOT USED	NOT USED
Max_Time_Between_Launches	Launch_Period(1)	NOT USED	NOT USED	NOT USED
Message_Ack_Operator	NOT USED	RPT	NOT USED	NOT USED
Message_Parameter_Name	PARAMETER_ NAME(1)	PARAMETER_ NAME(1)	PARAMETER_ NAME(1)	PARAMETER_ NAME(1)
Message_Request_Operator	RQCV	RQCV	RQCV	RQCV
Message_Update_Operator	LD	LD	LD	LD
Min_Time_Between_Launches	NOT USED	200 msec (±20 msec)	200 msec (±20 msec)	NOT USED
Threshold_For_Parameter_Change	NOT USED	change of 1 bit of resolution(2)	change of 1 bit of resolution(2)	NOT USED
Time_Between_Relaunches	NOT USED	850 msec (±100 msec)	NOT USED	NOT USED
Time_to_First_Relaunch	NOT USED	300 msec (±100 msec)	NOT USED	NOT USED

1. Value will be provided when the strategy is used in a process with a given parameter.
2. Value is typically provided when strategy is identified for the given parameter name.

6.8.2 Functional Requirements for Launch Message Template

The Launch Message Template provides the following functions:

a. Initiation of the launch of a message

b. Monitoring launches and ensuring the message is received across the network

c. Launching a message in response to a query message

d. Assigning a name and an operator to a message

Figure 43 provides an illustration of the data flow between the processes which accomplish these functions.

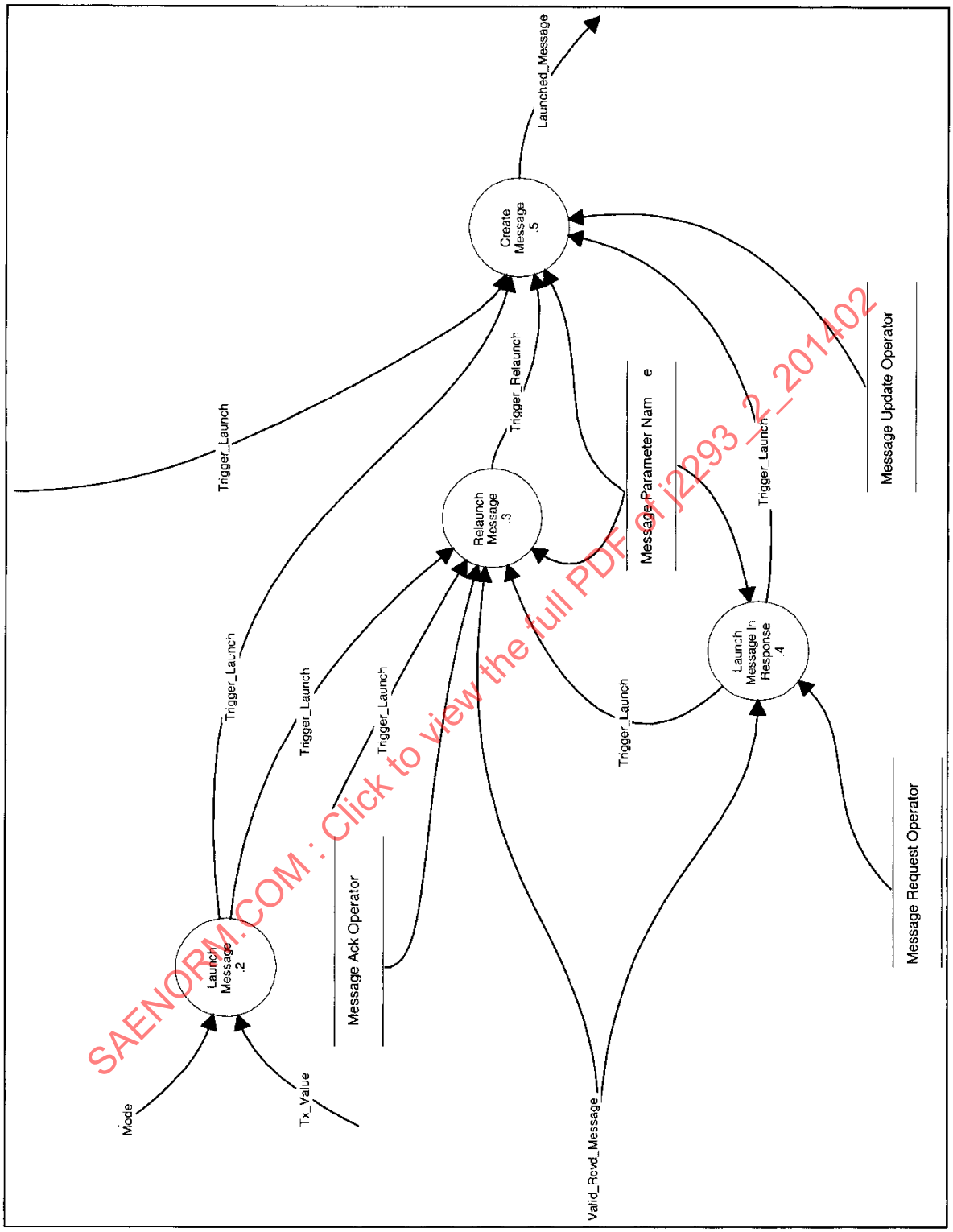

FIGURE 43 - LAUNCH MESSAGE TEMPLATE DFD

6.8.2.1 Process Control Requirements

None.

6.8.2.2 Launch Message

The launch message processes determines when to initiate a message launch to support one of the overall strategies for message launches. This process provides for launches which are initiated under the following circumstances:

a. Launched on process activation

b. Launched on mode change

c. Launched in parameter value change

d. Launched on a periodic basis

Additionally, this process provides for time spacing between message launches if desired.

Figure 44 provides an illustration of the data flow among the processes which accomplish these functions.

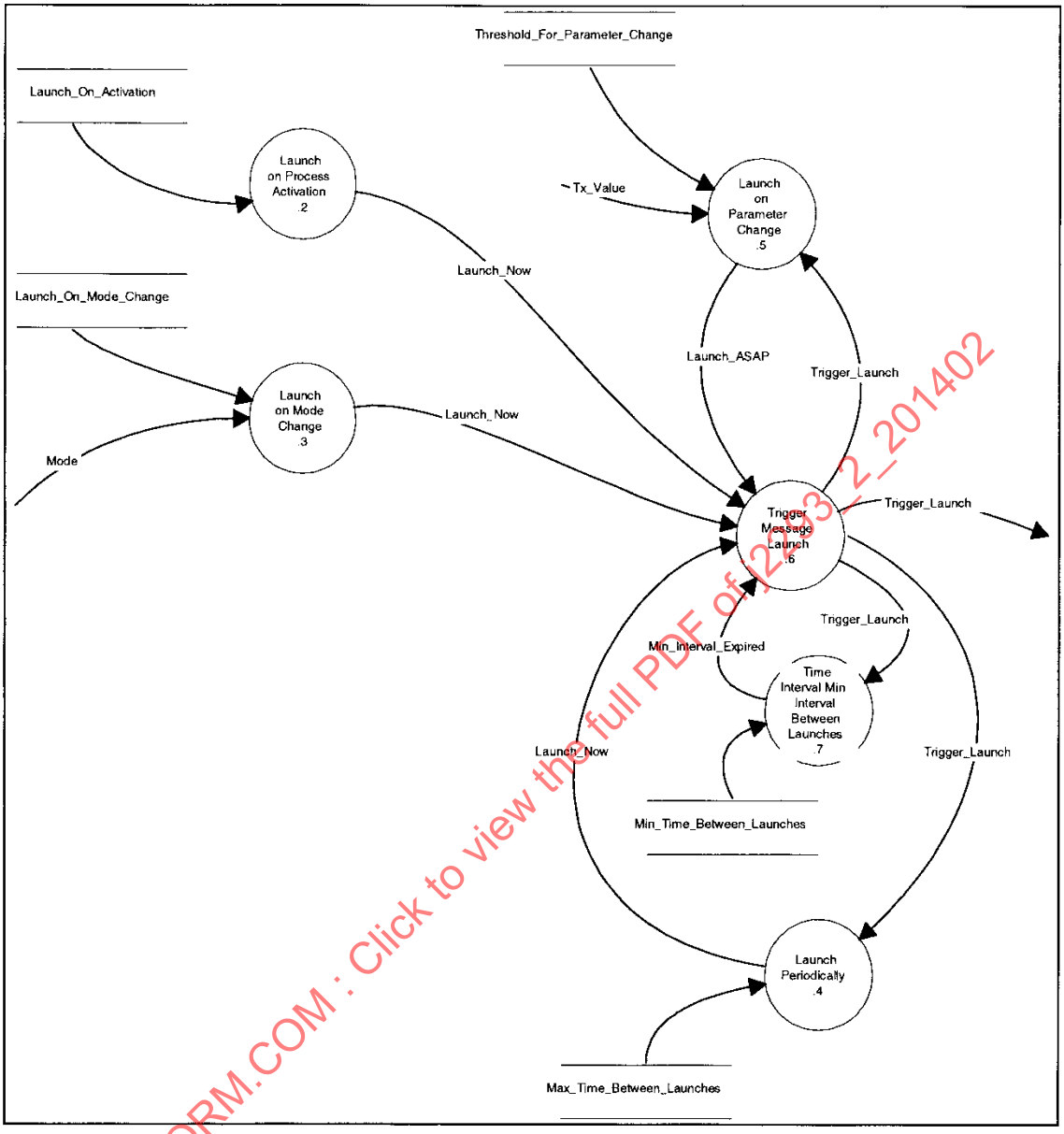

FIGURE 44 - LAUNCH MESSAGE DFD

6.8.2.2.1 Process Control Requirements

None.

6.8.2.2.2 P-spec—Launch on Process Activation

a. *Input/Output Flows*

Flow Name	*Type*
Launch_On_Activation	INPUT
Launch_Now	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process.

If Launch_On_Activation has a value of TRUE then
Issue Launch_Now with a value of TRUE.

c. *Operation*—The following shall be performed continually while this process is active:

None.

6.8.2.2.3 P-spec—Launch on Mode Change

a. Input/Output Flows

Flow Name	*Type*
Launch_On_Mode_Change	INPUT
Mode	INPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—The following shall be performed continually while this process is active:

If Launch_On_Mode_Change has a value of TRUE and the value of Mode has changed then
Issue Launch_Now with a value of TRUE.

6.8.2.2.4 P-spec: Launch Message Periodically

a. Input/Output Flows

Flow Name	*Type*
Trigger_Launch	INPUT
Max_Time_Between_Launches	INPUT
Launch_Now	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process.

None.

c. *Operation*—he following shall be performed continually while this process is active:

If Max_Time_Between_Launches contains a number and a period of time greater then
Max_Time_Between_Launches (±100 msec) has passed without Trigger_Launch having a value of TRUE,
Issue Launch_Now with a value of TRUE.

6.8.2.2.5 P-spec—Launch on Parameter Change

a. *Input/Output Flows*

Flow Name	*Type*
Tx_Value	INPUT
Threshold_For_Parameter_Change	INPUT
Trigger_Launch	INPUT
Launch_ASAP	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process.

None.

c. *Operation*—The following shall be performed continually while this process is active:

If Tx_Value has changed
If Threshold_For_Parameter_Change has a value and any changes in Tx_Value since either the last time that Trigger_Launch had a value of TRUE or since activation of this process meets the criteria in Threshold_For_Parameter_Change,
Issue Launch_ASAP with a value of TRUE.

6.8.2.2.6 P-spec—Trigger Message Launch

a. *Input/Output Flows*

Flow Name	*Type*
Launch_Now	INPUT
Launch_ASAP	INPUT
Min_Interval_Expired	INPUT
Trigger_Launch	OUTPUT

b. *Operation*—The following shall be performed continually while this process is active:

This process shall behave functionally equivalent to the State Transition Diagram shown in Figure 45.

6.8.2.2.7 P-spec—Detect Minimum Launch Interval Has Expired

a. *Input/Output Flows*

Flow Name	*Type*
Trigger_Launch	INPUT
Min_Interval_Between_Launches	INPUT
Min_Interval_Expired	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

Assign Min_Interval_Expired the value of TRUE.

c. *Operation*—The following shall be performed continually while this process is active:

If Min_Time_Between_Launches contains a number and more than Min_Time_Between_Launches seconds (±100 msecs) has passed without Trigger_Launch having a value of TRUE
  Assign Min_Interval_Expired the value of TRUE,
otherwise,
  Assign Min_Interval_Expired the value of FALSE.

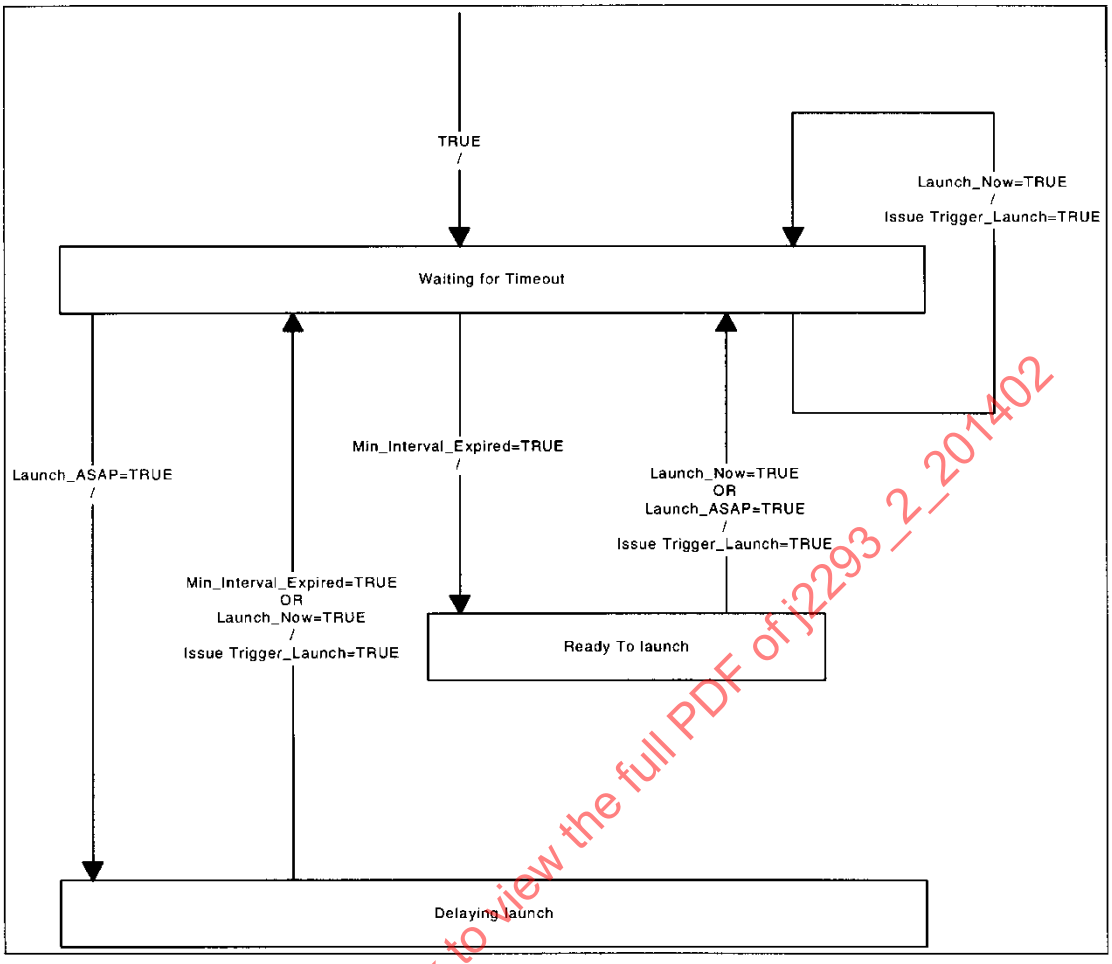

FIGURE 45 - DETERMINE LAUNCH STD

6.8.2.3 Relaunch Message

This group of processes provides the function of ensuring that a message is received across the network. Typically, the function provided by these processes will only be used with messages which are launched using a On Change strategy. One exception is where a system user may close the loop on confirming a message was successfully received. In messages which are used in this fashion, message relaunching would not be used. Figure 46 provides an illustration of the processes and data flows contained in this process.

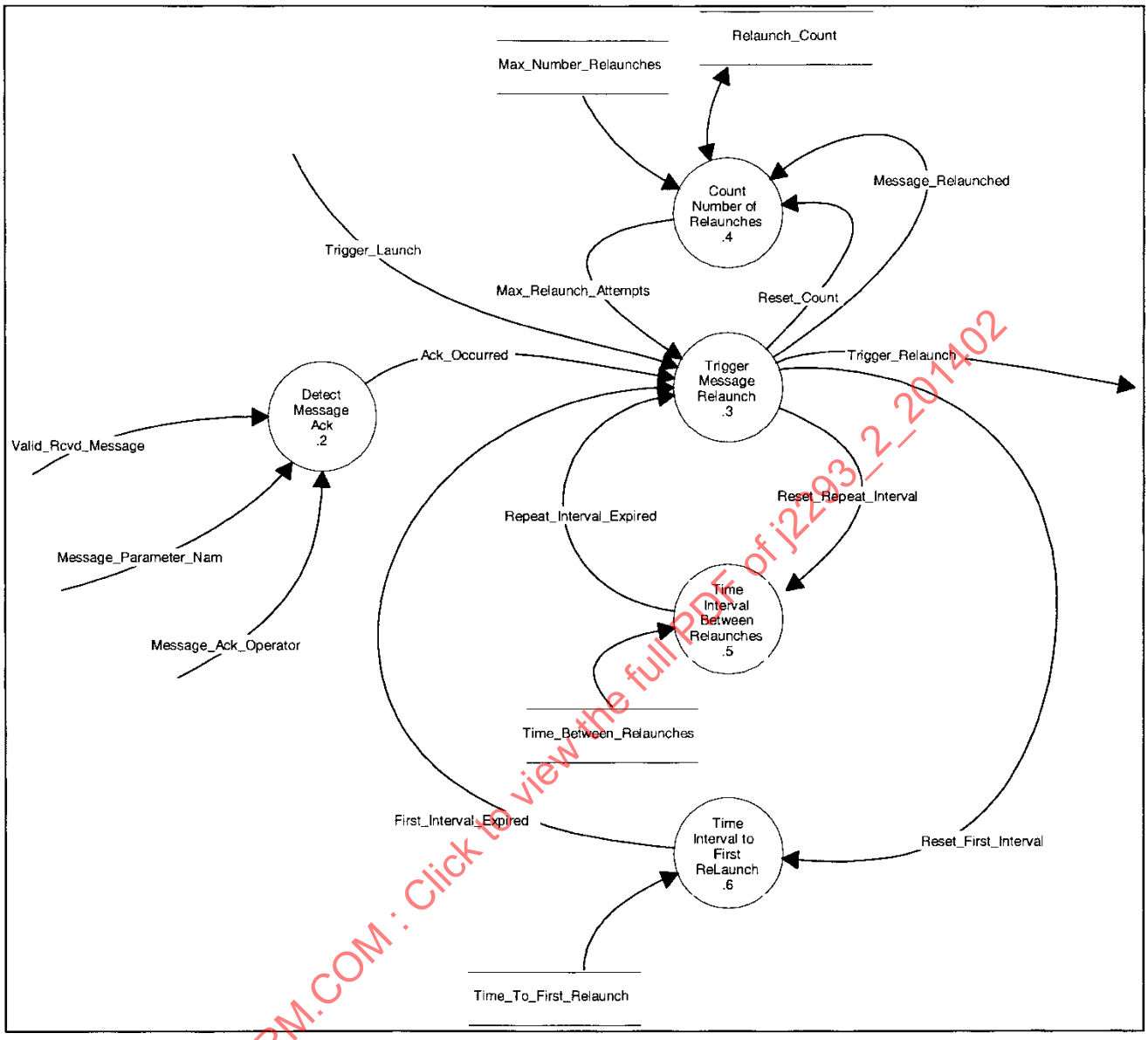

FIGURE 46 - RELAUNCH MESSAGE DFD

6.8.2.3.1 Process Control Requirements

None.

6.8.2.3.2 P-spec—Detect Message Ack

a. *Input/Output Flows*

Flow Name	*Type*
Valid_Rcvd_Message	INPUT
Message_Parameter_Name	INPUT
Message_Ack_Operator	INPUT
Ack_Occurred	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—The following shall be performed continually while this process is active:

If Valid_Rcvd_Message has a parameter name which is equal to the value of Message_Parameter_ Name and Valid_Rcvd_Message has an operator equal to the value of Message_Ack_Operator then
Issue Ack_Occurred with a value of TRUE.

6.8.2.3.3 P-spec—Trigger Message Relaunch

a. *Input/Output Flows*

Flow Name	*Type*
Trigger_Launch	INPUT
Max_Relaunch_Attempts	INPUT
Ack_Occurred	INPUT
First_Interval_Expired	INPUT
Repeat_Interval_Expired	INPUT
Max_Retry_Attempts	INPUT
Trigger_Relaunch	OUTPUT

b. *Operation*—he following shall be performed continually while this process is active:

This process shall behave functionally equivalent to the State Transition Diagram shown in Figure 47.

6.8.2.3.4 P-spec—Count Number of Message Relaunches

a. *Input/Output Flows*

Flow Name	*Type*
Max_Number_Relaunches	INPUT
Message_Relaunched	INPUT
Reset_Count	INPUT
Max_Relaunch_Attempts	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

Assign Max_Relaunch_Attempts the value of FALSE.
Assign Relaunch_Count the value of 0.

c. *Operation*

1. While this process is active, the following shall be performed on receipt of Reset_Count:

   If Reset_Count has a value of TRUE then
   Assign Max_Retry_Attempts the value of FALSE.
   Assign Relaunch_Count the value of 0.

2. While this process is active, the following shall be performed on receipt of Message_Relaunched:

   Increment Relaunch_Count by 1.
   if Max_Number_Of_Relaunches contains a number and Relaunch_Count is greater than or equal to Max_Number_Of_Relaunches then
   Assign Max_Relaunch_Attempts the value of TRUE,
   otherwise
   Assign Max_Relaunch_Attempts the value of FALSE.

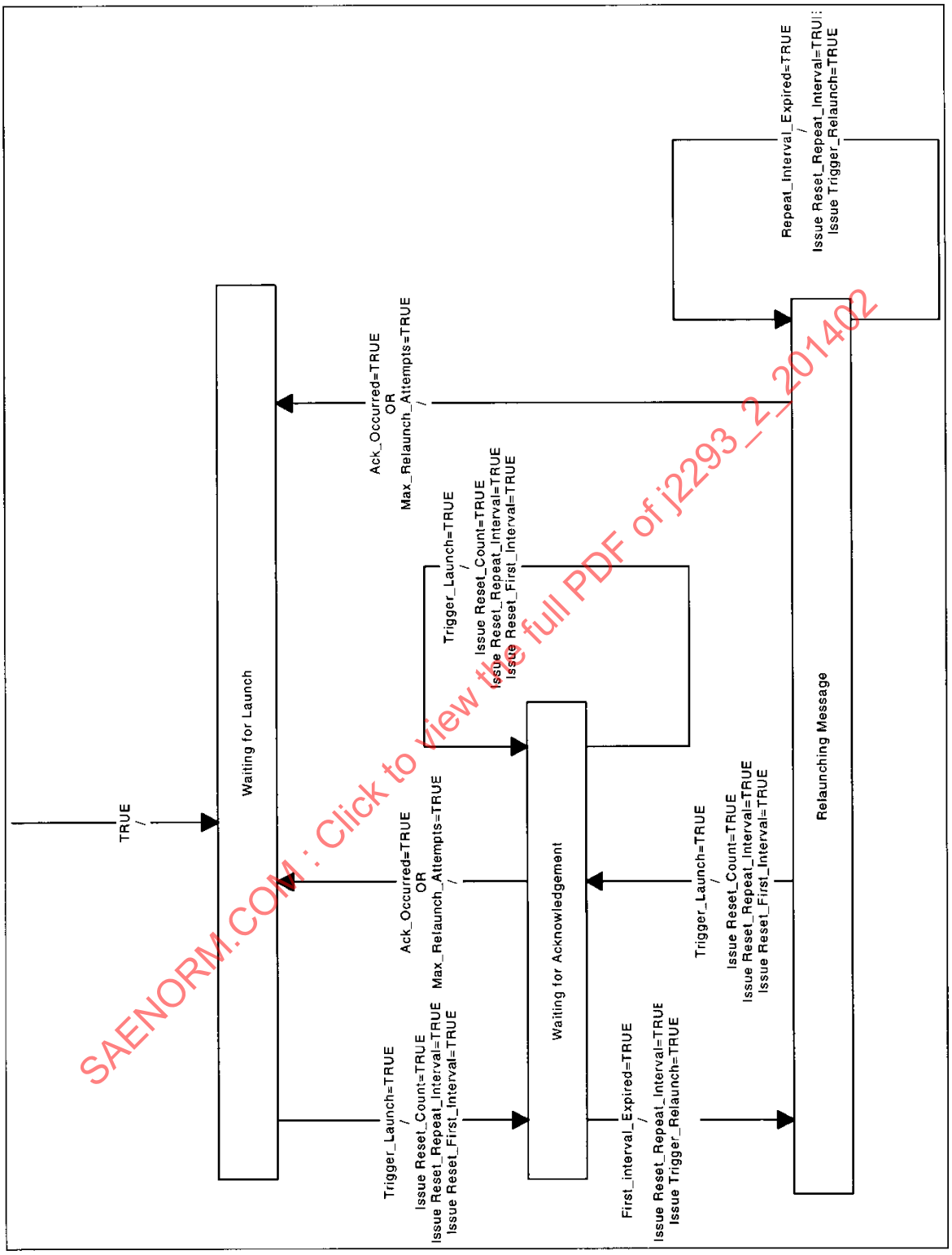

FIGURE 47 - DETERMINE WHEN TO RELAUNCH STD

6.8.2.3.5 P-spec—Time Interval Between Relaunches

a. *Input/Output Flows*

Flow Name	*Type*
Time_Between_Relaunches	INPUT
Reset_Repeat_Interval	INPUT
Repeat_Interval_Expired	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

Assign Repeat_Interval_Expired the value of FALSE.

c. *Operation*—The following shall be performed continually while this process is active:

If Time_Between_Relaunches contains a number and a period of time which is greater than or equal to Time_Between_Relaunches (±100 msec) has passed without Reset_Repeat_Interval having a value of TRUE then
  Assign Repeat_Interval_Expired the value of TRUE.
otherwise
  Assign Repeat_Interval_Expired the value of FALSE.

6.8.2.3.6 P-spec—Time Interval To First Relaunch

a. *Input/Output Flows*

Flow Name	*Type*
Time_To_First_Relaunch	INPUT
Reset_First_Interval	INPUT
First_Interval_Expired	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

Assign First_Interval_Expired the value of FALSE.

c. *Operation*—The following shall be performed continually while this process is active:

If Time_To_First_Relaunch contains a number and a period of time which is greater than or equal to Time_To_First_Relaunch (±100 msec) has passed without Reset_First_Interval having a value of TRUE then
  Assign First_Interval_Expired the value of TRUE,
otherwise
  Assign First_Interval_Expired the value of FALSE.

6.8.2.4 P-spec—Launch Message In Response

a. *Input/Output Flows*

Flow Name	*Type*
Valid_Rcvd_Message	INPUT
Message_Request_Operator	INPUT
Message_Parameter_Name	INPUT
Trigger_Launch	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—While this process is active, the following shall be performed on receipt of Valid_Rcvd_Message:

If the operator in Valid_Rcvd_Message is equal to the value of Message_Request_Operator and the parameter name in Valid Rcvd_Message is equal to the value of Message_Parameter_Name then
Issue Trigger_Launch with the value of TRUE.

6.8.2.5 P-spec—Create Message

a. *Input/Output Flows*

Flow Name	*Type*
Trigger_Launch	INPUT
Trigger_Relaunch	INPUT
Message_Update_Operator	INPUT
Message_Parameter_Name	INPUT
Launched_Message	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*

1. While this process is active, the following shall be performed on receipt of Trigger_Launch:

   If Trigger_Launch has a value of TRUE then
   Issue Launched_Message with a Parameter Name equal to the value of Message_Parameter_ Name and an Operator equal to the value of Message_Update_Operator.

2. While this process is active, the following shall be performed on receipt of Trigger_Relaunch:
   If Trigger_Relaunch has a value of TRUE then
   Issue Launched_Message with a Parameter Name equal to the value of Message_Parameter_ Name and an Operator equal to the value of Message_Update_Operator.

6.8.3 Data Dictionary for Launch Message Template

Table 45 contains the definitions for the Launch Message Template data and control flows. All flows are described using Backus-Naur form (BNF). Any element in the Data Dictionary which has an unassigned value will be referred to as having a NULL value in the document.

TABLE 45 - DATA DICTIONARY FOR LAUNCH MESSAGE TEMPLATE

NAME	Contained in / Consists of	Units	Range	Resolution	Description
Ack_Occurred	= [TRUE | FALSE]	—	—	—	
Byte_Data	In: Launched_Message Valid_Rcvd_Message	none	0 to 255	1	
First_Interval_Expired	= [TRUE | FALSE]	—	—	—	
Launch_ASAP	= [TRUE | FALSE]	—	—	—	
Launch_Now	= [TRUE | FALSE]	—	—	—	
Launch_On_Activation	= [TRUE | FALSE]	—	—	—	
Launch_On_Mode_Change	= [TRUE | FALSE]	—	—	—	
Launched_Message	Context Flow = (Parameter_Name) + (Operator) +(0 { Byte_Data } 12)	—	—	—	
Max_Number_Relaunches	= [number | UNLIMITED | NOT USED]	count	0 .. 6	1	
Max_Relaunch_Attempts	= [TRUE | FALSE]	—	—	—	
Max_Time_Between_Launches	= [number | NOT USED]	sec	0 .. 10	10 msec	
Message_Ack_Operator	= [RPT | NOT USED]	—	—	—	
Message_Parameter_Name	= Parameter_Name	—	—	—	
Message_Relaunched	= [TRUE | FALSE]	—	—	—	
Message_Request_Operator	= [RQCV | REQ | NOT USED]	—	—	—	
Message_Update_Operator	= [LD | RPT]	—	—	—	
Min_Interval_Expired	= [TRUE | FALSE]	—	—	—	
Min_Time_Between_Launches	= [number | NOT USED]	msec	0 .. 1000	100 msec	
Mode	Context Flow = [any mode defined in the context flow]	—	—	—	
Operator	In: Launched_Message Valid_Rcvd_Message = [LD | RPT | RQCV | REQ]	—	—	—	

TABLE 45 - DATA DICTIONARY FOR LAUNCH MESSAGE TEMPLATE (CONTINUED)

NAME	Contained in / Consists of	Units	Range	Resolution	Description
Parameter_Name	In: Launched_Message Message_Parameter_Name Valid_Rcvd_Message = [Name of any valid SAE J2178 Message]	—	—	—	
Relaunch_Count	= [number]	count	0 .. 6	1	
Repeat_Interval_Expired	= [TRUE | FALSE]	—	—	—	
Reset_Count	= [TRUE | FALSE]	—	—	—	
Reset_First_Interval	= [TRUE | FALSE]	—	—	—	
Reset_Repeat_Interval	= [TRUE | FALSE]	—	—	—	
Threshold_For_Parameter_Change	= [any desired description of a change in Tx_Value]	—	—	—	
Time_Between_Relaunches	= [number | NOT USED]	msec	0 .. 4000	100 msec	
Time_To_First_Relaunch	= [number | NOT USED]	msec	0 .. 4000	100 msec	
Trigger_Launch	Context Flow = [TRUE | FALSE]	—	—	—	
Trigger_Relaunch	= [TRUE | FALSE]	—	—	—	
Tx_Value	Context Flow = [any value defined for Tx_Value as it appears at the context]	—	—	—	
Valid_Rcvd_Message	Context Flow = (Parameter_Name) + (Operator) +(0 { Byte_Data { 12)	—	—	—	

6.9 Maintain Parameter Template

6.9.1 Context For Maintain Parameter Template

This template provides a set of requirements for maintaining a parameter over the network. These requirements define how to perform the following:

a. Receiving messages

b. Decoding data carried by the messages

c. How to recover from network errors

d. How to update and default data which is provided over the network

These functional requirements integrate the strategies defined in SAE J2178/5 and those needed to support the ETS network. Most messages will use a subset of these requirements to maintain the value of a parameter which is obtained from network traffic.

This template is applied throughout this document. First flows from the local context are substituted in place of context flows shown in Figure 48. Next, the read only data stores used throughout this template are assigned values to specify which strategies are used. Finally, a process activation table is configured to control when these sets of strategies are used. This method allows this template to describe any desired strategy or combination of strategies which is to be applied to a message.

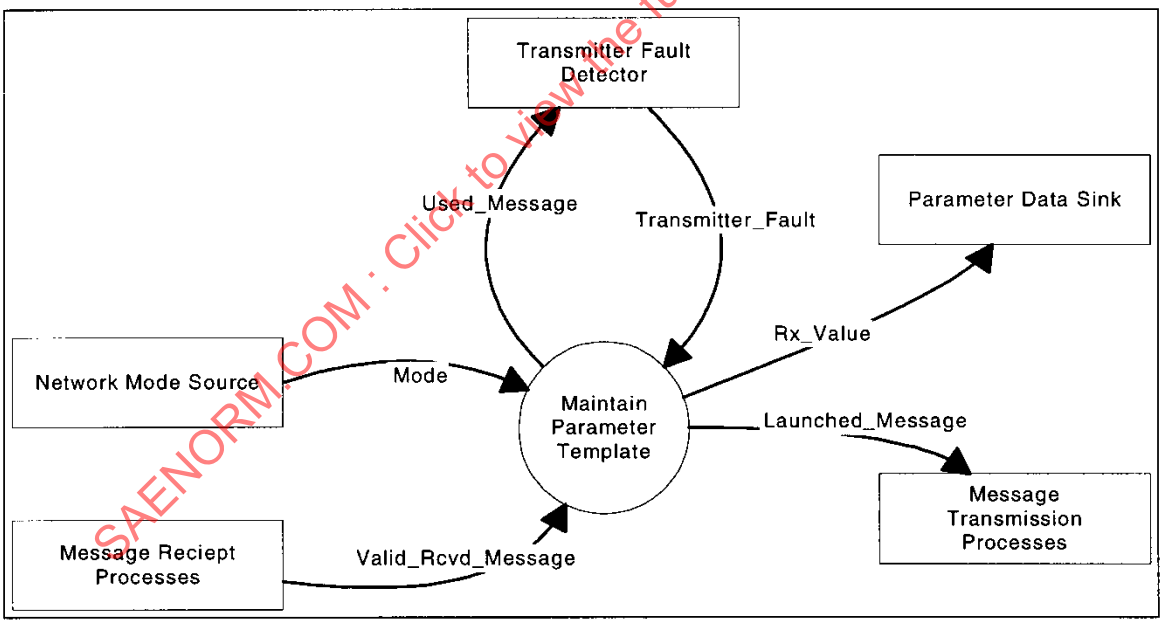

FIGURE 48 - MAINTAIN PARAMETER TEMPLATE FUNCTIONAL CONTEXT

Since this template is intended to contain all of the requirements related to maintaining a parameter, it also contains requirements for launching message requests and message acknowledgments. See Tables 46, 47 and 48.

TABLE 46 - SAE J2293-2 RECEPTION STRATEGIES

Parameter Maintenance Strategy Name	Parameter Maintenance Strategy
Receive On Change with Verification	A Message is expected periodically when the value in the transmitter changes. Upon receipt of a valid value, an acknowledgment is sent out. Recovery from errors in this strategy is linked to detection of a fault in a specific transmitter.
Receive On Change without Verification	A Message is expected aperiodically when the value in the transmitter changes. No acknowledgment is sent on receipt of a value. Recovery from errors in this strategy is linked to detection of a fault in a specific transmitter.
Receive On Initialization	Message is expected to be received during the initialization of a data channel or a change in application modes. If the message is not received by a specific time, a request for that message is launched. Recovery from errors is linked to detection of transmitter faults.
Periodic Receive	The message is expected to be received on a periodic basis. If the message is not received within a certain time, default actions occur. Recovery from errors in this strategy is linked to detection of faults on a specific message.

TABLE 47 - MAINTAIN PARAMETER TEPLATE CONFIGURATION STORES

Store Name	Description
Acknowledge_Updates	This flag is set to TRUE to configure the receiver to launch an acknowledgment that a message has been received and successfully processed.
Default_Value	This is a description of the value to issue to the Parameter Data Sink in the context under certain conditions. It describes the value that this parameter should have when the network can not establish a reliable value for this parameter.
Default_Value_on_Mode_Change	This flag is set to TRUE to indicate that Rx_Value should be issued with the Default_Value when the value of the Mode input at the Context changes.
Default_Value_on_Periodic_Fault	This flag is set to TRUE to indicate that the Rx_Value should be issued with the Default_Value when a message is not received on a periodic basis.
Default_Value_on_Process_Activation	This flag is set to TRUE to indicate that the Rx_Value should be issued with the Default_Value when this process is activated.
Default_Value_on_Process_Deactivation	This flag is set to TRUE to indicate that the Rx_Value should be issued with the Default_Value when this process is Deactivated.
Default_Value_on_Transmitter_Fault	This flag is set to TRUE to indicate that the Rx_Value should be issued with the Default_Value when a transmitter fault is detected for the source transmitter of any previous receptions of this message.
Detect_Update_Once	This flag is set to TRUE to indicate that the message is only expected once. If this value is set to TRUE, then after the message is received, all recovery strategies are disabled until the recovery strategy is reset.
Max_Number_Of_Requests	This value limits the maximum number requests which will be generated before the recovery process stops requesting a message.
Maximum_Update_Interval	For messages which are sent on a periodic basis, this value determines the maximum period of time which can pass before recovery and default strategies are enabled.
Message_Ack_Operator	see Table 43
Message_Parameter_Name	see Table 43
Message_Request_Operator	see Table 43
Message_Update_Operator	see Table 43
Request_Interval	Once a message recovery strategy has started to request messages, this value determines the interval between subsequent requests for transmission of a message.
Reset_Detection_on_Mode_Change	This flag is to TRUE to indicate that the message error recovery strategy is reset on an application mode change.

TABLE 48 - TYPICAL VALUES FOR RECEIVE STRATEGIES

Store Name	Periodic Receive Nominal Values	Receive on Change w/ Verification Nominal Values	Receive on Change w/o Verification Nominal Values	Receive on Initialization Nominal Values
Acknowledge_Updates	FALSE	TRUE	FALSE	FALSE
Default_Value	Default value as defined in Data Dictionary for parameters in message	default value as defined in Data Dictionary for parameters in message	default value as defined in Data Dictionary for parameters in message	default value as defined in Data Dictionary for parameters in message
Default_Value_on_Mode_Change	FALSE	FALSE	FALSE	FALSE
Default_Value_on_Periodic_Fault	TRUE	FALSE	FALSE	FALSE
Default_Value_on_Process_Activation	FALSE	FALSE	FALSE	FALSE
Default_Value_on_Process_Deactivation	TRUE	TRUE	TRUE	TRUE
Default_Value_on_Transmitter_Fault	FALSE	TRUE	TRUE	TRUE
Detect_Update_Once	FALSE	FALSE	FALSE	TRUE
Max_Number_Of_Requests	NOT USED	5	5	5
Maximum_Update_Interval	2.5 * PERIOD of launch strategy	NOT USED	NOT USED	NOT USED
Message_Ack_Operator	NOT USED	RPT	RPT	RPT
Message_Parameter_Name	Parameter_Name	Parameter_Name	Parameter_Name	Parameter_Name
Message_Request_Operator	NOT USED	RQCV	RQCV	RQCV
Message_Update_Operator	LD	LD	LD	LD
Request_Interval	NOT USED	500 msec	500 msec	500 msec
Reset_Detection_on_Mode_Change	FALSE	TRUE	TRUE	TRUE

6.9.2 Functional Requirements for Maintain Parameter Template

The Maintain Parameter Template provides the following functions which:

a. Detect when a message which updates the monitored parameter is received

b. Decode a received message into data which is passed on to other processes

c. Attempt to recovery a value from a transmitter during detected fault conditions

d. Detect when to default a parameter's value

e. Acknowledge that the receiver has successfully decoded a message

Figure 49 provides an illustration of the data flow among the processes which accomplish these functions.

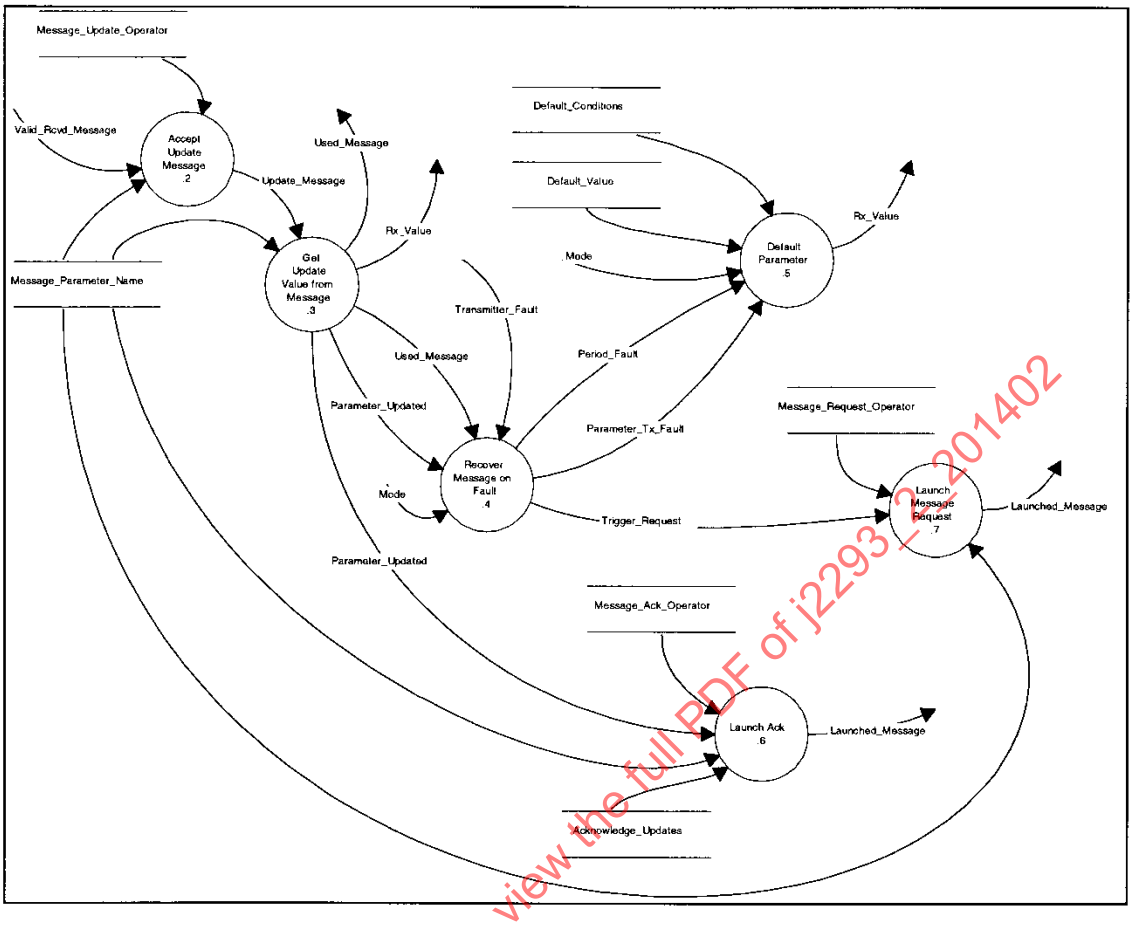

FIGURE 49 - MAINTAIN PARAMETER TEMPLATE DFD

6.9.2.1 Process Control Requirements

None.

6.9.2.2 P-spec—Accept Update Message

a. *Input/Output Flows*

Flow Name	*Type*
Valid_Rcvd_Message	INPUT
Message_Update_Operator	INPUT
Message_Parameter_Name	INPUT
Update_Message	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—While this process is active, the following shall be performed on receipt of Valid_Rcvd_Message:

If the operator in Valid_Rcvd_Message is equal to the value of Message_Update_Operator and the parameter name in Valid_Rcvd_Message is equal to the value of Message_Parameter_Name then
Issue Update_Message with the value of Valid_Rcvd_Message.

6.9.2.3 P-spec—Get Update Value from Message

a. *Input/Output Flows*

Flow Name	*Type*
Update_Message	INPUT
Message_Parameter_Name	
Used_Message	OUTPUT
Rx_Value	OUTPUT
Parameter_Update	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—While this process is active, the following shall be performed on receipt of Valid_Rcvd_Message:

If the data fields in Update_Message can be decoded and turned into values as defined in SAE J2178 for the name in Message_Parameter_Name then
Issue Rx_Value with this decoded value, and
Issue Used_Message with the value of Update_Message, and
Issue Parameter_Updated with the value of TRUE.

6.9.2.4 Recover Parameter on Detected Fault

This group of processes determines

a. When to launch a request message

b. Which transmitter is associated with this message

c. When a message has a periodic fault

d. When a message has a transmitter fault

Figure 50 illustrates the relation between these processes.

6.9.2.4.1 P-spec—Process Control Requirements

None.

6.9.2.4.2 P-spec—Record Message Source Address

a. *Input/Output Flows*

Flow Name	*Type*
Mode	INPUT
Reset_Detection_On_Mode_Change	INPUT
Used_Message	INPUT
Message_Source_Address	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

Clear Message_Source_Address.

c. *Operation*—While this process is active, the following shall be performed on receipt of Used_Message:

Assign Message_Source_Address the value of Byte 3 in Used_Message.

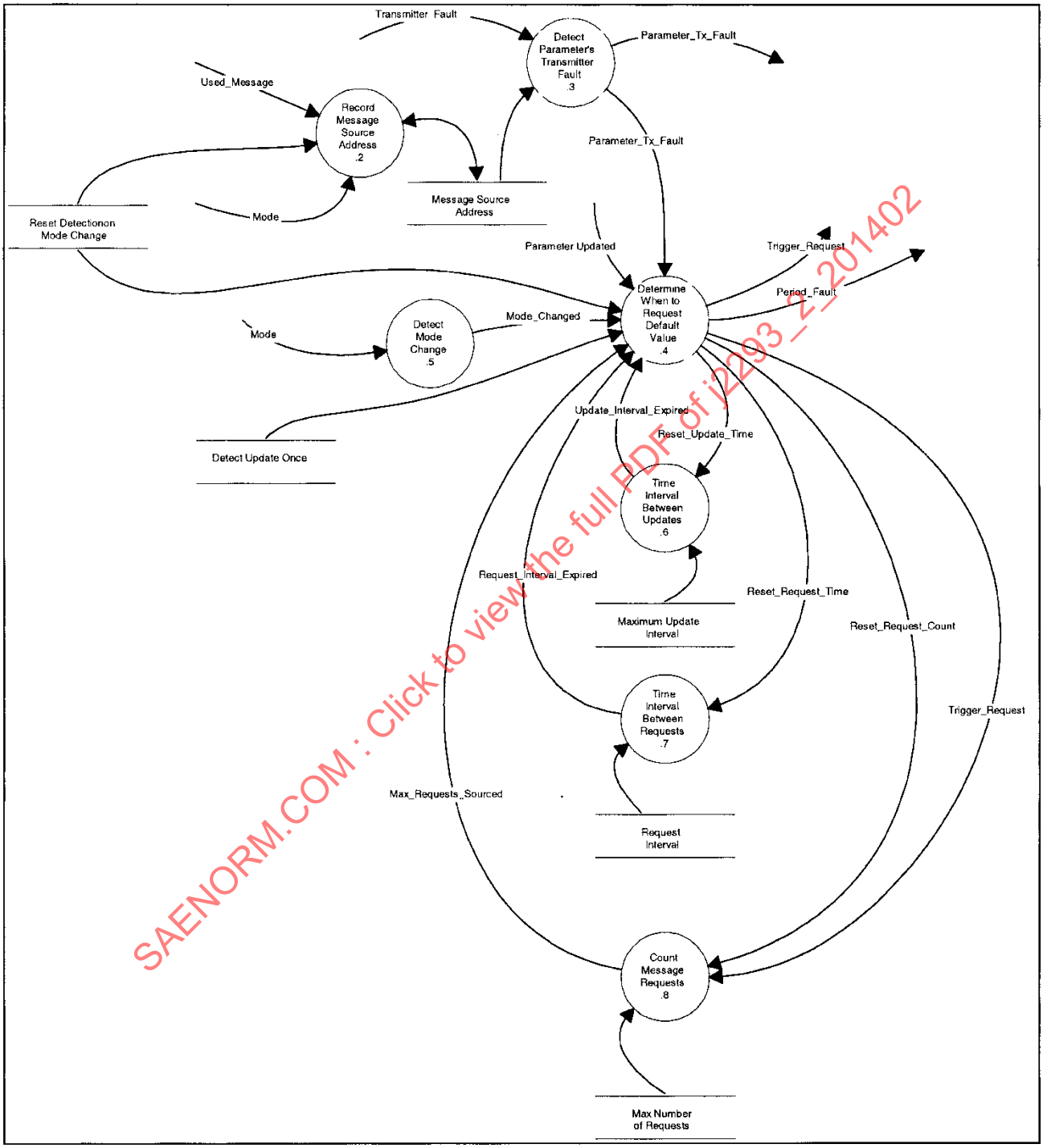

FIGURE 50 - RECOVER PARAMETER ON FAULT DFD

6.9.2.4.3 P-spec—Detect Parameter's Transmitter Fault

a. *Input/Output Flows*

Flow Name	*Type*
Transmitter_Fault	INPUT
Message_Source_Address	INPUT
Parameter_Tx_Fault	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

Assign Parameter_Tx_Fault the value of FALSE.

c. *Operation*—The following shall be performed continually while this process is active:

If the any of the values contained in Transmitter_Fault are equal to the value in Message_Source_Address then
  Assign Parameter_Tx_Fault the value of TRUE,
otherwise,
  Assign Parameter_Tx_Fault the value of FALSE.

6.9.2.4.4 P-spec—Determine when to Request Default Value

a. *Input/Output Flows*

Flow Name	*Type*
Parameter_Tx_Fault	INPUT
Parameter_Updated	INPUT
Mode_Changed	INPUT
Detect_Update_Once	INPUT
Reset_Detection_On_Mode_Change	INPUT
Update_Interval_Expired	INPUT
Request_Interval_Expired	INPUT
Max_Requests_Sourced	INPUT
Trigger_Request	OUTPUT
Period_Fault	OUTPUT
Reset_Request_Count	OUTPUT
Reset_Request_Interval	OUTPUT
Reset_Update_Interval	OUTPUT

b. *Operation*—The following shall be performed continually while this process is active:

This process shall behave functionally equivalent to the State Transition Diagram shown in Figure 49. See Figure 51.

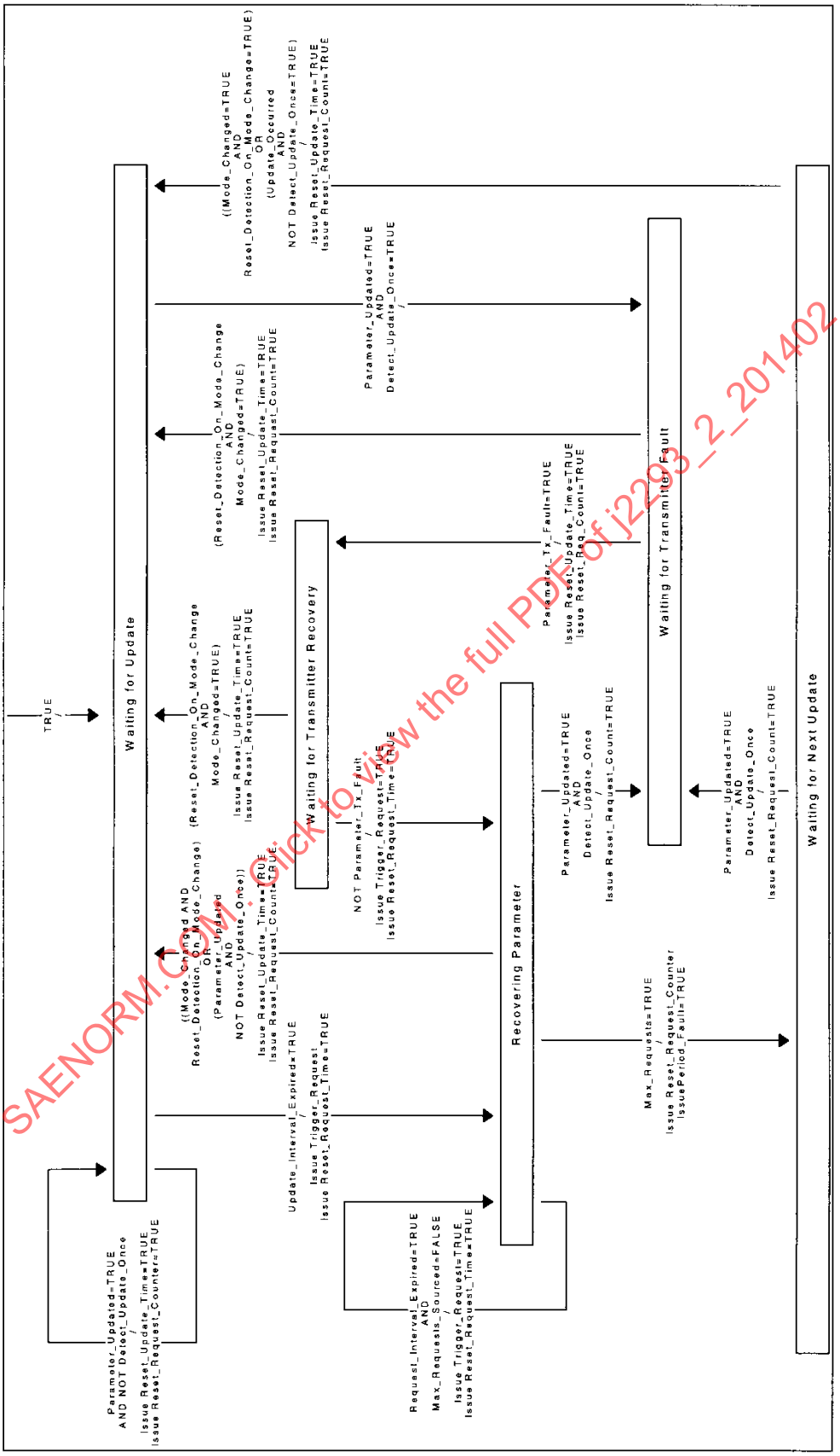

FIGURE 51 - DETERMINE WHEN TO REQUEST DEFAULT VALUE STD

6.9.2.4.5 P-spec—Detect Mode Change

a. *Input/Output Flows*

Flow Name	*Type*
Mode	INPUT
Mode_Changed	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—The following shall be performed continually while this process is active:

If the value of Mode changes then
Issue Mode_Changed with a value of TRUE.

6.9.2.4.6 P-spec—Time Interval Between Updates

a. *Input/Output Flows*

Flow Name	*Type*
Reset_Update_Interval	INPUT
Maximum_Update_Interval	INPUT
Update_Interval_Expired	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

Assign Update_Interval_Expired the value of FALSE.

c. *Operation*—The following shall be performed continually while this process is active:

If Maximum_Update_Interval contains a number and a period of time which is greater than or equal to Maximum_Update_Interval (±100 msec) has passed without Reset_Update_Interval having a value of TRUE then,
  Assign Update_Interval_Expired the value of TRUE,
otherwise,
  Assign Update_Interval_Expired the value of FALSE.

6.9.2.4.7 P-spec—Time Interval Between Requests

a. *Input/Output Flows*

Flow Name	*Type*
Request_Interval_Expired	INPUT
Reset_Request_Interval	INPUT
Request_Interval_Expired	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

Assign Request_Interval_Expired the value of FALSE.

c. *Operation*—The following shall be performed continually while this process is active:

If Request_Interval contains a number and a period of time which is greater than or equal to Request_Interval (±100 msec) has passed without Reset_Request_Interval having a value of TRUE then,
  Assign Request_Interval_Expired the value of TRUE,
otherwise,
  Assign Request_Interval_Expired the value of FALSE.

6.9.2.4.8 P-spec—Count Message Requests

a. *Input/Output Flows*

Flow Name	*Type*
Reset_Request_Count	INPUT
Trigger_Request	INPUT
Max_Number_Of_Requests	INPUT
Max_Requests_Sourced	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

Assign Max_Requests_Sourced the value of FALSE.

c. *Operation*—The following shall be performed continually while this process is active:

If the number of times Trigger_Request has had the value of TRUE since the most recent of either process activation or Reset_Request_Count having a value of TRUE is equal to Max_Number_Of_Requests then
  Assign Max_Requests_Sourced the value of TRUE,
otherwise,
  Assign Max_Requests_Sourced the value of FALSE.

6.9.2.4.9 Storage Requirements

The parent process contain storage for the following dataflows. These storage values are read only storage. The value for each of these stores are assigned when this templates use is specified.

a. Detect_Update_Once

b. Reset_Detection_On_Mode_Change

c. Maximum_Update_Interval

d. Request_Interval

e. Max_Number_Of_Requests

6.9.2.5 P-spec—Default Parameter

a. *Input/Output Flows*

Flow Name	*Type*
Default_Value	INPUT
Default_Value_On_Mode_Change	INPUT
Default_Value_On_Periodic_Fault	INPUT
Default_Value_On_Process_Activation	INPUT
Default_Value_On_Process_Deactivation	INPUT
Default_Value_On_Transmitter_Fault	INPUT
Period_Fault	INPUT
Parameter_Tx_Fault	INPUT
Mode	INPUT
Rx_Value	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

If Default_On_Process_Activation has a value of TRUE then
Issue Rx_Value with the value of Default_Value.

c. *Operation*—The following shall be performed continually while this process is active:

When any of the event identified in Table 49 occurs, if the conditions in the Enabling Conditions column evaluate to TRUE, then
Issue Rx_Value with the value of Default_Value.

TABLE 49 - DEFAULTING EVENTS

Event	Enabling Conditions
Period_Fault has a value of TRUE	Default_Value_On_Periodic_Fault = TRUE
Parameter_Tx_Fault has a value of TRUE	Default_Value_On_Transmitter_Fault = TRUE
the value of Mode changes	Default_Value_On_Mode_Change = TRUE

d. *Deactivation*—The following shall be performed once immediately prior to deactivation of this process:

If Default_On_Process_Deactivation has a value of TRUE then
Issue Rx_Value with the value of Default_Value.

6.9.2.6 P-spec—Launch Message Request

a. *Input/Output Flows*

Flow Name	*Type*
Trigger_Request	INPUT
Message_Request_Operator	INPUT
Message_Parameter_Name	INPUT
Launched_Message	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—While this process is active, the following shall be performed on receipt of Trigger_Request:

If Trigger_Request has a value of TRUE then
Issue Launched_Message with
a Parameter Name equal to Message_Parameter_Name,
and an Operator equal to Message_Request_Operator.

6.9.2.7 P-spec—Launch Ack

a. *Input/Output Flows*

Flow Name	*Type*
Message_Ack_Operator	INPUT
Parameter_Updated	INPUT
Message_Parameter_Name	INPUT
Acknowledge_Updates	INPUT
Launched_Message	OUTPUT

b. *Initialization*—The following shall be performed once upon activation of this process:

None.

c. *Operation*—While this process is active, the following shall be performed on receipt of Parameter_Updated:

If Parameter_Updated has a value of TRUE then
Issue Launched_Message with
a Parameter Name equal to Message_Parameter_Name,
and an Operator equal to Message_Ack_Operator.

6.9.2.8 Storage Requirements

The parent process contain storage for the following dataflows. These storage values are read only storage. The value for each of these stores are assigned when this templates use is specified.

a. Message_Update_Operator

b. Message_Parameter_Name

c. Default_Conditions

d. Default Value

e. Message_Request_Operator

f. Message_Ack_Operator

g. Acknowledge_Updates

6.9.3 Data Dictionary For Maintain Parameter Template

Table 50 contains the definitions for the Maintain Parameter Template data and control flows. All flows are described using Backus-Naur form (BNF). Any element in the Data Dictionary which has an unassigned value will be referred to as having a NULL value in the document.

TABLE 50 - MAINTAIN PARAMETER TEMPLATE DATA DICTIONARY

NAME	Contained in / Consists of	Units	Range	Resolution	Description
Parameter_Name	In: Launched_Message Update_Message Used_Message Valid_Rcvd_Message = [Name of any valid SAE J2178 Message]	—	—	—	
Operator	In: Launched_Message Update_Message Used_Message Valid_Rcvd_Message = [LD | RPT | RQCV | REQ]	—	—	—	
Byte_Data	In: Launched_Message Update_Message Used_Message Valid_Rcvd_Message	none	0 to 255	1	
Reset_Request_Time	= [TRUE | FALSE]	—	—	—	
Maximum_Update_Interval		msec	0 to 4000	10	
Reset_Update_Time	= [TRUE | FALSE]	—	—	—	
Request_Number		none	0 to 7	1	
Acknowledge_Updates	= [TRUE | FALSE]	—	—	—	
Default_Conditions	= Default_Value_On_Mode_Change + Default_Value_On_Process_Activation + Default_Value_On_Process_Deactivation + Default_Value_On_Transmitter_Fault	—	—	—	
Default_Value	= [any single value which is valid for Rx_Value at the context]	—	—	—	
Default_Value_On_Mode_Change	= [TRUE | FALSE]	—	—	—	
Default_Value_On_Process_Activation	= [TRUE | FALSE]	—	—	—	
Default_Value_On_Process_Deactivation	= [TRUE | FALSE]	—	—	—	
Default_Value_On_Transmitter_Fault	= [TRUE | FALSE]	—	—	—	
Detect_Update_Once	= [TRUE | FALSE]				
Launched_Message	Context Flow = (Parameter Name) + (Operator) + (Data) + 4 { byte data } 11	—	—	—	

TABLE 50 - MAINTAIN PARAMETER TEMPLATE DATA DICTIONARY(CONTINUED)

NAME	Contained in / Consists of	Units	Range	Resolution	Description
Max_Number_Of_Requests	= [Request_Number | NOT_USED]	—	0 to 7	1	
Max_Requests_Sourced	= [TRUE | FALSE]				
Message_Ack_Operator	= [RPT | NOT USED]				
Message_Parameter_Name	= [any parameter name which has a PRN used for a secondary ID in SAE J2178]	—	—	—	
Message_Request_Operator	= [REQ | RQCV | NOT USED]	—	—	—	
Message_Source_Address		—	0 to 255	1	
Message_Update_Operator	= [LD | RPT | NOT USED]	—	—	—	
Minimum_Update_Interval	= [TRUE | FALSE]				
Mode	Context Flow = [any one of the value defined for this flow at the context]	—	—	—	
Mode_Changed	= [TRUE | FALSE]				
Parameter_Tx_Fault	= [TRUE | FALSE]				
Parameter_Updated	= [TRUE | FALSE]				
Period_Fault	= [TRUE | FALSE]				
Request_Interval		msec	0 to 4000	10	
Request_Interval_Expired	= [TRUE | FALSE]				
Reset_Detection_On_Mode_Change	= [TRUE | FALSE]				
Reset_Request_Count	= [TRUE | FALSE]				
Reset_Request_Interval	= [TRUE | FALSE]				
Reset_Update_Interval	= [TRUE | FALSE]				
Rx_Value	Context Flow = any value defined for this flow at the context	—	—	—	
Transmitter_Fault	Context Flow = 0 { Message Source Address }	—	0 .. 255	1	
Trigger_Request	= [TRUE | FALSE]				
Update_Interval_Expired	= [TRUE | FALSE]				
Update_Message	= (Parameter Name) + (Operator) + (Data) + 4 { Byte_Data } 11	—	—	—	

TABLE 50 - MAINTAIN PARAMETER TEMPLATE DATA DICTIONARY(CONTINUED)

NAME	Contained in / Consists of	Units	Range	Resolution	Description
Used_Message	Context Flow = (Parameter Name) + (Operator) + (Data) + 4 { byte data } 11	—	—	—	
Valid_Rcvd_Message	Context Flow = (Parameter Name) + (Operator) + (Data) + 4 { byte data } 11	—	—	—	

6.10 ETS Network Message Requirements

The requirements for the messages used by the ETS Network are identified in

a. Table 54,

b. Table 55, and

c. Table 56.

The messages and the calibrations to configure them are generated directly from the data dictionary in SAE J2293-1. The process used to generate these tables is defined in and uses the following tables and figures:

a. Table 51,

b. Table 52,

c. Table 53,

d. Figure 53,

e. Figure 54, and

f. Figure 55.

The message latency goals and mapping to J2293-1 dataflows are provided by Table 56. Since the transport of data may involve gateways or other sources of delay, three latencies are identified. Nominal Dataflow latency is the nominal time that it takes for data to be output from one process until it is received by another process. Nominal Interface Latency is the amount of Nominal Dataflow Latency which is allocated to the interface between the ETS Network and the processes which create and use the information. This is included to provide guidelines for maximum latency when gateways or other potential sources of significant delay are introduced. Nominal communication latency is the maximum delay the this ETS message should encounter when all applications which use the ETS Network are active. See Figures 52 and 53.

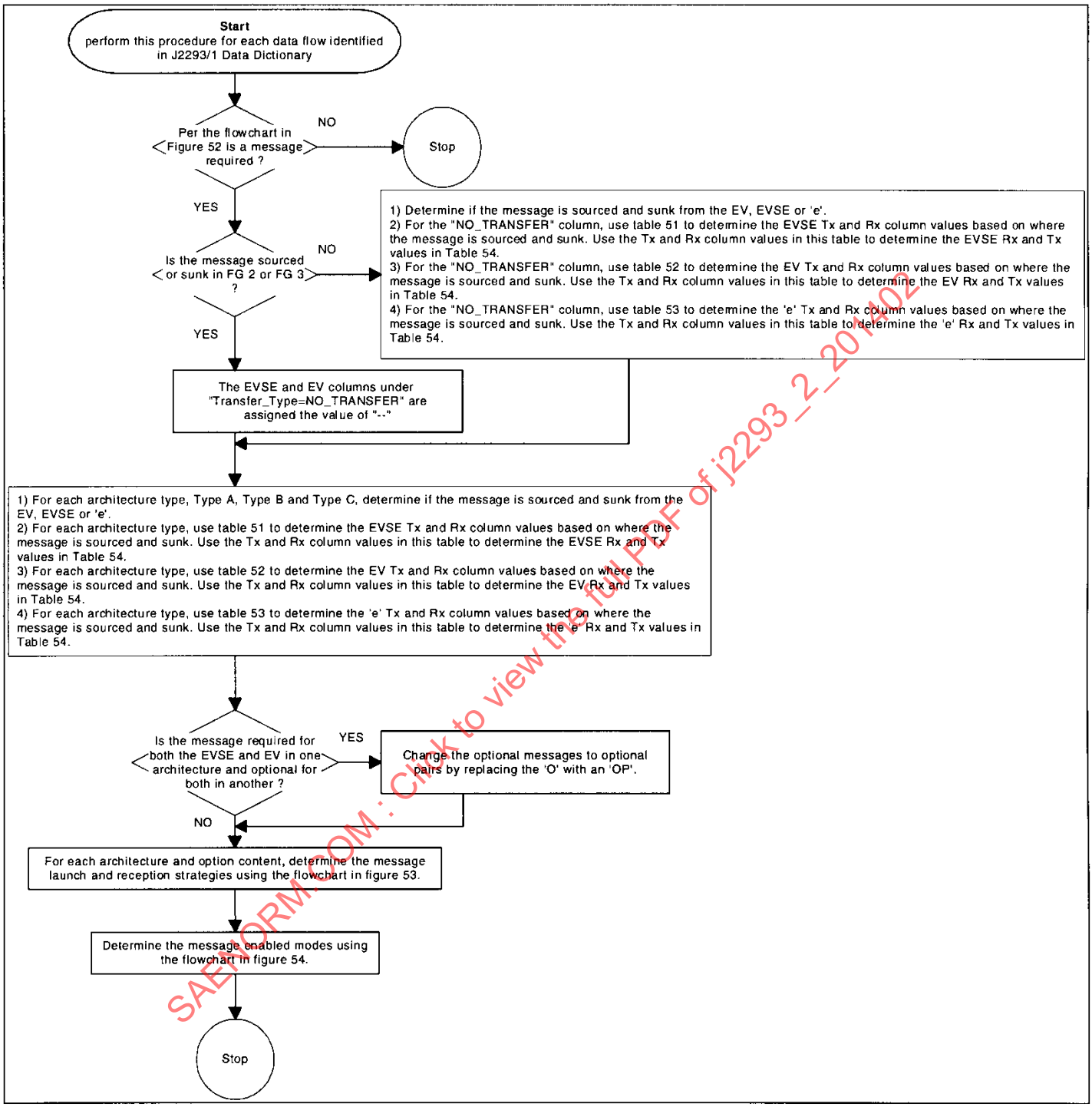

FIGURE 52 - FLOWCHART FOR CREATING MESSAGE TABLES

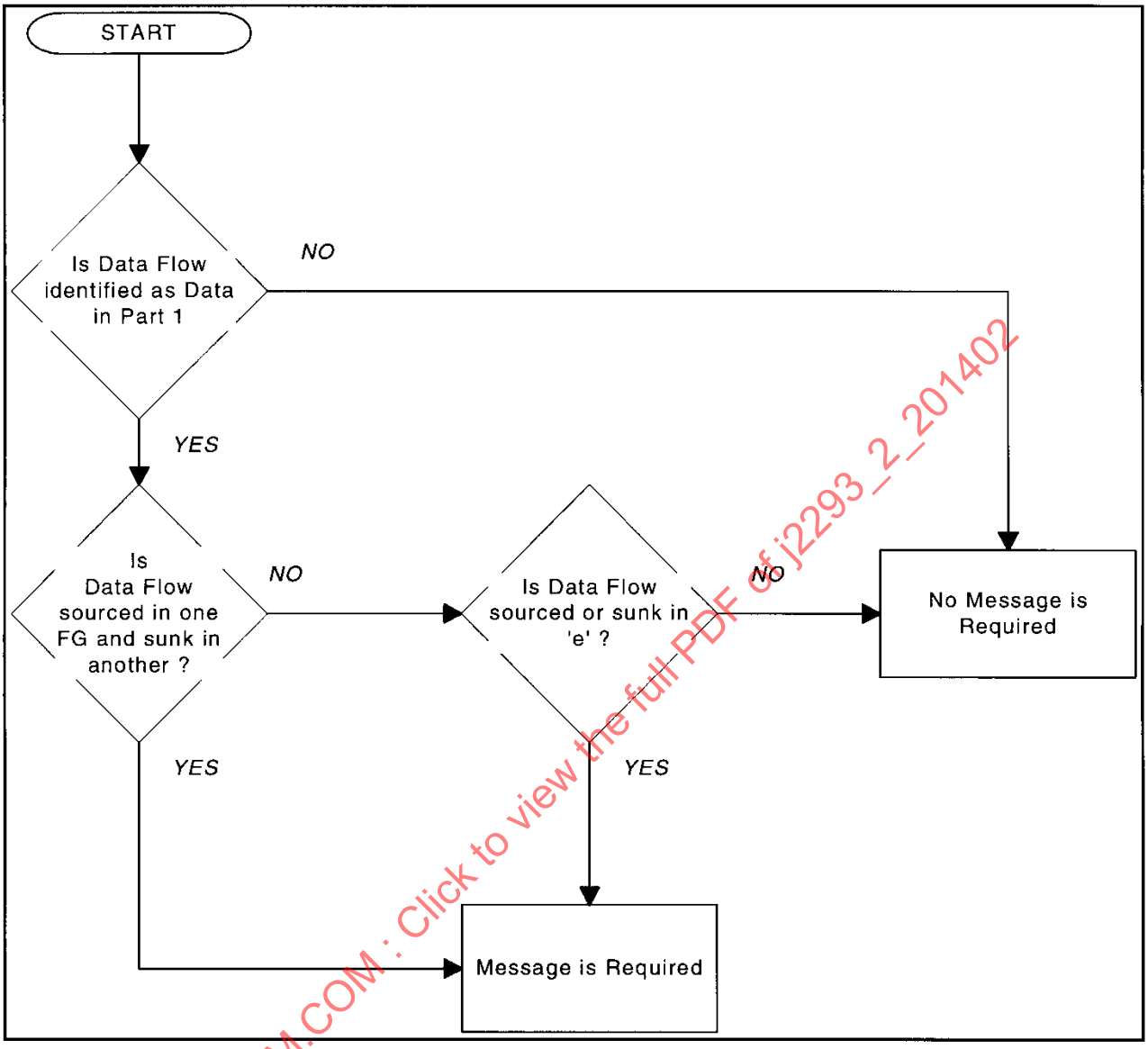

FIGURE 53 - FLOWCHART FOR SELECTION OF DATA FLOWS FOR MESSAGES

TABLE 51 - EVSE MESSAGE REQUIREMENT MATRIX

Data Flow Routing for Architecture and Mode[1] in Use EVSE to EVSE	Data Flow Routing for Architecture and Mode[1] in Use EVSE to EV	Data Flow Routing for Architecture and Mode[1] in Use EVSE to EV	Data Flow Routing for Architecture and Mode[1] in Use EVSE to EV	Data Flow Routing for Architecture and Mode[1] in Use EVSE to EV	EVSE Tx	EVSE Rx	Verification used Rx
NO	NO	NO	NO	NO	—	—	—
NO	NO	NO	NO	YES	R	—	NO
NO	NO	NO	YES	NO	R	—	YES
NO	NO	NO	YES	YES	R	—	YES
NO	NO	YES	NO	NO	—	O	NO
NO	NO	YES	NO	YES	X	X	X
NO	NO	YES	YES	NO	X	X	X
NO	NO	YES	YES	YES	X	X	X
NO	YES	NO	NO	NO	—	R	YES
NO	YES	NO	NO	YES	X	X	X
NO	YES	NO	YES	NO	X	X	X
NO	YES	NO	YES	YES	X	X	X
NO	YES	YES	NO	NO	—	R	YES
NO	YES	YES	NO	YES	X	X	X
NO	YES	YES	YES	NO	X	X	X
NO	YES	YES	YES	YES	X	X	X
YES	NO	NO	NO	NO	O	O	YES
YES	NO	NO	NO	YES	R	O	NO
YES	NO	NO	YES	NO	R	O	YES
YES	NO	NO	YES	YES	R	O	YES
YES	NO	YES	NO	NO	X	X	X
YES	NO	YES	NO	YES	X	X	X
YES	NO	YES	YES	NO	X	X	X
YES	NO	YES	YES	YES	X	X	X
YES	YES	NO	NO	NO	X	X	X
YES	YES	NO	NO	YES	X	X	X
YES	YES	NO	YES	NO	X	X	X
YES	YES	NO	YES	YES	X	X	X
YES	YES	YES	NO	NO	X	X	X
YES	YES	YES	NO	YES	X	X	X
YES	YES	YES	YES	NO	X	X	X
YES	YES	YES	YES	YES	X	X	X

Legend: R = Required; O = Optional; — = Not Permitted; X = Illegal Combination

1. Modes are identified and illustrated in Table 4.

TABLE 52 - EVSE MESSAGE REQUIREMENT MATRIX

Data Flow Routing for Architecture and Mode(1) in Use EV to EV	Data Flow Routing for Architecture and Mode(1) in Use EVSE to EV	Data Flow Routing for Architecture and Mode(1) in Use 'e' to EV	Data Flow Routing for Architecture and Mode(1) in Use EV to EVSE	Data Flow Routing for Architecture and Mode(1) in Use EV to 'e'	EV Tx	EV Rx	Verification used Rx
NO	NO	NO	NO	NO	—	—	—
NO	NO	NO	NO	YES	R	—	NO
NO	NO	NO	YES	NO	R	—	YES
NO	NO	NO	YES	YES	R	—	YES
NO	NO	YES	NO	NO	—	O	NO
NO	NO	YES	NO	YES	X	X	X
NO	NO	YES	YES	NO	X	X	X
NO	NO	YES	YES	YES	X	X	X
NO	YES	NO	NO	NO	—	R	YES
NO	YES	NO	NO	YES	X	X	X
NO	YES	NO	YES	NO	X	X	X
NO	YES	NO	YES	YES	X	X	X
NO	YES	YES	NO	NO	—	R	YES
NO	YES	YES	NO	YES	X	X	X
NO	YES	YES	YES	NO	X	X	X
NO	YES	YES	YES	YES	X	X	X
YES	NO	NO	NO	NO	O	O	YES
YES	NO	NO	NO	YES	R	O	NO
YES	NO	NO	YES	NO	R	O	YES
YES	NO	NO	YES	YES	R	O	YES
YES	NO	YES	NO	NO	X	X	X
YES	NO	YES	NO	YES	X	X	X
YES	NO	YES	YES	NO	X	X	X
YES	NO	YES	YES	YES	X	X	X
YES	YES	NO	NO	NO	X	X	X
YES	YES	NO	NO	YES	X	X	X
YES	YES	NO	YES	NO	X	X	X
YES	YES	NO	YES	YES	X	X	X
YES	YES	YES	NO	NO	X	X	X
YES	YES	YES	NO	YES	X	X	X
YES	YES	YES	YES	NO	X	X	X
YES	YES	YES	YES	YES	X	X	X

Legend: R = Required; O = Optional; — = Not Permitted; X = Illegal Combination

1. Modes are identified and illustrated in Table 4. Architecture for the ETS system is identified in SAE J2293-1.

TABLE 53 - EXTERNAL MESSAGE REQUIREMENTS MATRIX

Data Flow Routing for Architecture and Mode[1] in Use 'e' is a source	Data Flow Routing for Architecture and Mode[1] in Use 'e' is a sink	External Tx	External Rx
NO	NO		
YES	NO	O	
NO	YES		O
YES	YES	X	X

Legend: R = Required; O = Optional; — = Not Permitted; X = Illegal Combination

1. M odes are identified and illustrated in Table 4. Architecture for the ETS system is identified in SAE J2293-1.