INSO
14848
1st. Edition
Jun.2013

جمهوری اسلامی ایران
Islamic Republic of Iran
سازمان ملی استاندارد ایران
Iran National Standardization Organization

استاندارد ملی ایران

۱۴۸۴۸

چاپ اول

خرداد ۱۳۹۲

فناوری اطلاعات - فراخوانی رویه مستقل از زبان (LIPC)

Information Technology - Language-Independent Procedure Calling (LIPC)

ICS: 35.060

به نام خدا

آشنایی با سازمان ملی استاندارد ایران

مؤسسهٔ استاندارد و تحقیقات صنعتی ایران به موجب بند یک مادهٔ۳ قانون اصلاح قوانین و مقررات مؤسسهٔ استاندارد و تحقیقات صنعتی ایران، مصوب بهمن ماه ۱۳۷۱ تنها مرجع رسمی کشور است که وظیفه تعیین، تدوین و نشر استانداردهای ملی (رسمی) ایران را به عهده دارد.

نام موسسه استاندارد و تحقیقات صنعتی ایران به موجب یکصد و پنجاه و دومین جلسه شورای عالی اداری مورخ ۹۰/۶/۲۹ به سازمان ملی استاندارد ایران تغییر و طی نامه شماره ۲۰۶/۳۵۸۳۸ مورخ ۹۰/۷/۲۴ جهت اجرا ابلاغ شده است .

تدوین استاندارد در حوزه های مختلف در کمیسیون های فنی مرکب از کارشناسان سازمان ، صاحب نظران مراکز و مؤسسات علمی، پژوهشی، تولیدی و اقتصادی آگاه و مرتبط انجام می شود وکوششی همگام با مصالح ملی و با توجه به شرایط تولیدی، فناوری و تجاری است که از مشارکت آگاهانه و منصفانهٔ صاحبان حق و نفع، شامل تولیدکنندگان، مصرف‌کنندگان، صادرکنندگان و وارد کنندگان، مراکز علمی و تخصصی، نهادها، سازمان های دولتی و غیر دولتی حاصل می شود .پیش نویس استانداردهای ملی ایران برای نظرخواهی به مراجع ذی نفع و اعضای کمیسیون های فنی مربوط ارسال می‌شود و پس از دریافت نظرها و پیشنهادها در کمیتهٔ ملی مرتبط با آن رشته طرح و در صورت تصویب به عنوان استاندارد ملی (رسمی) ایران چاپ و منتشر می شود.

پیش نویس استانداردهایی که مؤسسات و سازمان های علاقه مند و ذی صلاح نیز با رعایت ضوابط تعیین شده تهیه می کنند درکمیتهٔ ملی طرح و بررسی و درصورت تصویب، به عنوان استاندارد ملی ایران چاپ و منتشر می شود .بدین ترتیب، استانداردهایی ملی تلقی می شوند که بر اساس مفاد نوشته شده در استاندارد ملی ایران شمارهٔ ۵ تدوین و در کمیتهٔ ملی استاندارد مربوط که سازمان ملی استاندارد ایران تشکیل می‌دهد به تصویب رسیده باشد.

سازمان ملی استاندارد ایران از اعضای اصلی سازمان بین المللی استاندارد (ISO)[1] ،کمیسیون بین المللی الکتروتکنیک (IEC)[2] و سازمان بین المللی اندازه شناسی قانونی (OIML)[3] است و به عنوان تنها رابط[4] کمیسیون کدکس غذایی (CAC)[5] در کشور فعالیت می کند. در تدوین استانداردهای ملی ایران ضمن توجه به شرایط کلی و نیازمندی های خاص کشور ، از آخرین پیشرفت های علمی، فنی و صنعتی جهان و استانداردهای بین‌المللی بهره‌گیری می‌شود .

سازمان ملی استاندارد ایران می تواند با رعایت موازین پیش بینی شده در قانون، برای حمایت از مصرف کنندگان، حفظ سلامت و ایمنی فردی و عمومی، حصول اطمینان از کیفیت محصولات و ملاحظات زیست محیطی و اقتصادی، اجرای بعضی از استانداردهای ملی ایران را برای محصولات تولیدی داخل کشور و/یا اقلام وارداتی، با تصویب شورای عالی استاندارد، اجباری نماید. سازمان می تواند به منظور حفظ بازارهای بین المللی برای محصولات کشور، اجرای استاندارد کالاهای صادراتی و درجه‌بندی آن را اجباری نماید. همچنین برای اطمینان بخشیدن به استفاده کنندگان از خدمات سازمان ها و مؤسسات فعال در زمینهٔ مشاوره، آموزش، بازرسی، ممیزی و صدورگواهی سیستم های مدیریت کیفیت و مدیریت زیست‌محیطی، آزمایشگاه ها و مراکز کالیبراسیون (واسنجی) وسایل سنجش، سازمان ملی استاندارد ایران این گونه سازمان ها و مؤسسات را بر اساس ضوابط نظام تأیید صلاحیت ایران ارزیابی می کند و در صورت احراز شرایط لازم، گواهینامهٔ تأیید صلاحیت به آن ها اعطا و بر عملکرد آن ها نظارت می کند. ترویج دستگاه بین المللی یکاها، کالیبراسیون (واسنجی) وسایل سنجش، تعیین عیار فلزات گرانبها و انجام تحقیقات کاربردی برای ارتقای سطح استانداردهای ملی ایران از دیگر وظایف این سازمان است.

---

1 - International Organization for Standardization
2 - International Electrotechnical Commission
3 - International Organization of Legal Metrology (Organisation Internationale de Metrologie Legale)
4 - Contact point
5 - Codex Alimentarius Commission

کمیسیون فنی تدوین استاندارد
«فناوری اطلاعات – فراخوانی رویه مستقل از زبان(LIPC) »

رئیس:	**سمت و/ یا نمایندگی**
فولادیان، مجید (فوق لیسانس مهندسی مخابرات)	مشاور سازمان فناوری اطلاعات ایران
دبیر:	
میراسکندری، سید محمدرضا (کارشناس ارشد مهندسی کامپیوتر-نرم افزار)	مدیر کل خدمات ارزش افزوده سازمان فناوری اطلاعات
اعضاء: (اسامی به ترتیب حروف الفبا)	
بختیاری، شیرین (لیسانس مهندسی برق)	کارشناس تدوین استاندارد سازمان فناوری اطلاعات ایران
جمیل پناه، ناصر (کارشناس ارشد مدیریت)	کارشناس سازمان فناوری اطلاعات
سعیدی، عذراء (فوق لیسانس مهندسی مخابرات)	کارشناس تدوین استاندارد سازمان فناوری اطلاعات ایران
عبداللهی ازگمی، محمد (دکتری مهندسی کامپیوتر-نرم‌افزار)	استادیار دانشگاه علم و صنعت ایران
سلطانی حقیقت، الهه (لیسانس مهندسی برق مخابرات)	کارشناس سازمان فناوری اطلاعات ایران
فرهاد شیخ احمد، لیلا (فوق لیسانس مهندسی کامپیوتر نرم افزار)	کارشناس تدوین استاندارد سازمان فناوری اطلاعات ایران
فیاضی، مهدی (لیسانس مهندسی الکترونیک)	کارشناس مسؤول تدوین استاندارد و امنیت شبکه
قسمتی، سیمین (فوق لیسانس فناوری اطلاعات، لیسانس مهندسی الکترونیک)	کارشناس سازمان فناوری اطلاعات ایران

کبیری، پیمان (دکتری کامپیوتر)	استادیار دانشگاه علم و صنعت ایران
مجاهدی، الناز (لیسانس مهندسی کامپیوتر نرم افزار)	نماینده دانشگاه علم و صنعت ایران
معروف، سینا (لیسانس مهندسی کامپیوتر سخت‌افزار)	کارشناس سازمان فناوری اطلاعات ایران
میرزایی رضایی، طیبه (فوق لیسانس فیزیک)	رئیس اداره تدوین استاندارد ها و نظارت بر فرآیند سرویس‌ها سازمان فناوری اطلاعات
ناظمی، اسلام (دکتری کامپیوتر)	استادیار دانشگاه شهید بهشتی

فهرست مندرجات

پیش‌گفتار

استاندارد «فناوری اطلاعات –فراخوانی رویه مستقل از زبان (LIPC)» که پیش نویس آن درکمیسیون‌های مربوط به توسط سازمان فناوری اطلاعات تهیه و تدوین شده و در دویست و هفتم اجلاس کمیتهٔ ملی استاندارد رایانه و فرآوری داده مورخ ۹۱/۸/۲۸ مورد تصویب قرار گرفته است، اینک به استناد بند یک مادهٔ ۳ قانون اصلاح قوانین و مقررات سازمان استاندارد و تحقیقات صنعتی ایران، مصوب بهمن ماه ۱۳۷۱، به عنوان استاندارد ملی ایران منتشر می‌شود.

برای حفظ همگامی و هماهنگی با تحولات و پیشرفت‌های ملی و جهانی در زمینهٔ صنایع، علوم و خدمات، استانداردهای ملی ایران در مواقع لزوم تجدید نظر خواهد شد و هر پیشنهادی که برای اصلاح و تکمیل این استانداردها ارائه شود، هنگام تجدید نظر در کمیسیون فنی مربوط مورد توجه قرار خواهد گرفت. بنابراین، باید همواره از آخرین تجدیدنظر استانداردهای ملی استفاده کرد.

منبع و ماخذی که برای تهیهٔ این استاندارد مورد استفاده قرار گرفته به شرح زیر است:

ISO/IEC 13886:1996, Information technology - Language-Independent Procedure Calling (LIPC)

مقدمه

هدف از تدوین این استاندارد ملی، ارائه‌ی مدل مشترکی برای مفهوم فراخوانی رویه در استانداردهای زبان است. این استاندارد برای کمک به توسعه‌ی ابزارها و خدمات مستقل از زبان، کتابخانه‌های رروبه عمومی و برنامه‌نویسی چندزبانه است. در کاربردهای چندزبانه، رویه‌های کارساز[1] روی پردازنده‌های زبانی که در طریقه[2] کارساز عمل می‌کنند، اجرا می‌شوند و این رویه‌ها به وسیله پردازنده‌هایی که در طریقه کارخواه[3] عمل می‌کنند فراخوانی می‌شوند. توجه داشته باشید که لزومی ندارد زبان‌ها متفاوت باشند، و در صورت یکسان بودن پردازنده‌ها، این مدل به برنامه‌نویسی تک پردازه‌ای متعارف تنزل پیدا می‌کند.

اغلب زبان‌های برنامه‌نویسی، مفاهیم رویه‌ها و فراخوانی آن‌ها را دارند. تفاوت اصلی بین روش‌های استفاده شده در زبان‌های برنامه‌نویسی مختلف در روشی است که برای ارسال پارامترها بین رویه‌های کارخواه و کارساز استفاده می‌کنند. فراخوانی رویه در سطح عملکردی مفهوم ساده‌ای است، اما تعامل فراخوانی رویه با نوع داده، ساختار برنامه، گونه‌های مختلف فراخوانی رویه و محدودیت‌های فراخوانی که به وسیله زبان‌های برنامه‌نویسی مختلف اعمال می‌شوند، مفهوم به ظاهر ساده‌ی فراخوانی رویه را به یک ویژگی پیچیده زبان‌های برنامه‌نویسی تبدیل می‌کند.

شکل‌های مختلف فراخوانی رویه در زبان‌های استاندارد، نشان‌دهنده‌ی نیاز به یک مدل استاندارد برای فراخوانی رویه است. وجود این استاندارد ملی برای فراخوانی رویه مستقل از زبان (LIPC)[4] به این معنی نیست که تمام زبان‌های برنامه‌نویسی باید فقط از این مدل به عنوان تنها وسیله‌ی فراخوانی رویه استفاده کنند. اما زبان‌های برنامه نویسی حداقل نیاز دارند که نگاشتی از سازوکار فراخوانی رویه خودشان به LIPC داشته باشند و بتوانند فراخوانی‌های زبان‌های برنامه‌نویسی دیگری که نگاشتی به این استاندارد ملی تعریف کرده‌اند را بپذیرند.

این استاندارد ملی مشخصات یک مدل مشترک برای فراخوانی رویه است و قصد ندارد مشخصات چگونگی پیاده‌سازی LIPC را بیان کند. همچنین باید توجه داشته باشیم که این استاندارد به سؤالاتی نظیر نحوه‌ی برقراری ارتباط فراخوانی رویه (که به وسیله پردازنده طریقه کارخواه آغاز شده است) با پردازنده طریقه کارساز، یا چگونگی بازگرداندن نتایج پاسخ نمی‌دهد. مدل تعریف شده در این استاندارد ملی قرار است به وسیله زبان‌ها به شکلی استفاده شود که بتوانند نگاشت‌های استانداردی از مدل رویه ذاتی خودشان ارائه کنند. این استاندارد ملی به استاندارد ملی انواع داده‌ی مستقل از زبان[5] (ISO/IEC 11404) جهت تعریف انواع داده‌ای که قرار است در مدل LIPC ارائه شده پشتیبانی شوند، وابسته است.

---

1 - Server
2 - Mode
3 - Client
4 - Language-Independent Procedure Calling
5 - Language-Independent Datatypes

فناوری اطلاعات - فراخوانی رویه مستقل از زبان (LIPC)

۱ هدف و دامنه کاربرد

هدف از تدوین این استاندارد ملی، مشخص کردن مدل فراخوانی رویه‌ها، و نحوِ[1] مرجع برای نگاشت «به» و «از» مدل است. به این نحو، نشانه‌گذاری تعریف واسط (IDN)[2] گفته می‌شود. مدل تعریف شده شامل ویژگی‌هایی از قبیل فراخوانی رویه، ارسال پارامتر، وضعیت[3] تکمیل و مسائل محیطی مربوط به مرجع‌های غیرمحلی و حالت[4] می‌شود.

این استاندارد ملی برای موارد زیر کاربرد ندارد:

- روشی که فراخوانی رویه آغاز شده به وسیله پردازنده طریقه کارخواه، برای برقراری ارتباط با پردازنده زبان طریقه کارساز استفاده می‌کند.
- کمینه الزامات سامانه‌ی پردازش داده‌ای که قادر به پشتیبانی از یک پیاده‌سازی پردازنده زبان برای پشتیبانی از LIPC است.
- سازوکاری که به وسیله آن برنامه‌های نوشته شده برای پشتیبانی LIPC، جهت استفاده‌ی یک سامانه‌ی پردازش داده تغییر پیدا می‌کند.
- نمایش[5] یک پارامتر

یادآوری - در ابتدا قصد داشتیم تا تعاریف و مفاهیم این استاندارد ملی با استاندارد فراخوانی رویه راه دور (RPC)[6] (ISO/IEC 11578) در یک راستا باشند. اما متأسفانه در مرحله‌ی آخر فرآیند توسعه، تصمیم گرفته شد تا برای این استاندارد از رویکرد به‌طور کامل متفاوتی استفاده شود. لذا هم‌راستایی مورد نظر عملی نشد. «پیوست ت» مروری بر تفاوت‌های مفاهیم تعریف شده‌ی این استاندارد ملی و استاندارد RPC دارد.

۲ مراجع الزامی

مدارک الزامی زیر حاوی مقرراتی است که در متن این استاندارد ملی ایران به آنها ارجاع داده شده است. بدین ترتیب آن مقررات جزئی از این استاندارد ملی ایران محسوب می شود.

در صورتی که به مدرکی با ذکر تاریخ انتشار ارجاع داده شده باشد، اصلاحیه‌ها و تجدید نظرهای بعدی آن مورد نظر این استاندارد ملی نیست. در مورد مدارکی که بدون ذکر تاریخ انتشار به آن‌ها ارجاع داده شده است، همواره تاریخ تجدید نظر و اصلاحیه‌های بعدی آن‌ها مورد نظر است.

استفاده از مراجع زیر برای این استاندارد الزامی است:

2-1 ISO 2375:1985, *Data processing - Procedure for registration of escape sequences*.

---

1 - Syntax
2 - Interface Definition Notation
3 - Status
4 - State
5 - Representation
6 - Remote Procedure Call

2-2 ISO/IEC 10646-1:1993, *Information technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane.*

2-3 ISO/IEC 11404:1996, *Information technology - Programming languages, their environment and system software interfaces - Language-independent datatypes.*

2-4 ISO/IEC 8824-1:1995, *Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

2-5 ISO/IEC 8825-1:1995, *Information technology -ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).*

۳ اصطلاحات، تعاریف و کوته‌نوشت‌ها

در این استاندارد اصطلاحات و تعاریف زیر به کار می‌رود:

۳-۱- تعاریف

در این استاندارد، تعاریف زیر به‌کار رفته است:

۳-۱-۱

پارامتر واقعی[۱]

مقداری که در زمان اجرای یک رویه، به یک پارامتر صوری[۲] مقید[۳] می‌شود.

۳-۱-۲

همبستگی[۴]

هر نگاشتی از یک مجموعه از نمادها به مقادیر است.

۳-۱-۳

جعبه[۵]

مدلی از یک متغیر یا دربردارنده[۶]، که مقداری از یک نوع خاص را نگه می‌دارد.

۳-۱-۴

انقیاد واسط کارخواه[۷]

مالکیت یک مرجع واسط به وسیله رویه کارخواه است.

1 - Actual parameter
2 - Formal
3 - Bound
4 - Association
5 - Box
6 - Container
7 - Client interface binding

۳-۱-۵

رویه کارخواه[1]

دنباله‌ای[2] از دستورالعمل‌هایی که رویه دیگری را فراخوانی می‌کند.

۳-۱-۶

بستار رویه کامل[3]

بستار رویه‌ی که تمام نمادهای سراسری آن نگاشته می‌شوند.

۳-۱-۷

پیکربندی[4]

رایانه‌های میزبان و مقصد، سامانه‌های عامل و نرم‌افزار به کارگرفته شده برای اجرای یک پردازنده است.

۳-۱-۸

دنباله اجرا[5]

یک توالی از حالت‌های سراسری s_1، s_2 و ... است، که در آن هر حالت به جز حالت اول، از حالت قبلی خودش با یک تک عملیات ایجاد[6] یا یک تک عملیات نوشتن[7] مشتق می‌شود.

۳-۱-۹

پارامتر صوری[8]

نماد اسمی پارامتر به کار گرفته شده در تعریف رویه است و در زمان اجرا به آن مقداری مقید خواهد شد.

۳-۱-۱۰

حالت سراسری[9]

مجموعه‌ای از تمام جعبه‌های موجود و مقادیر کنونی تخصیص یافته به آن‌ها است.

۳-۱-۱۱

نماد سراسری[10]

نمادی که برای ارجاع به مقادیری که همواره با یک رویه، همبسته هستند به کار گرفته می‌شود.

---

1 - Client procedure
2 - Sequence
3 - Complete procedure closure
4 - Configuration
5 - Execution sequence
6 - Create
7 - Write
8 - Formal Parameter
9 - Global state
10 - Global symbol

۳-۱-۱۲

تعریف‌شده در پیاده‌سازی[1]

ویژگی تعریف شده در پیاده‌سازی، ویژگی است که در این استاندارد ملی به پیاده‌سازی واگذار می‌شود. اما هرگونه پیاده‌سازی مدعی انطباق با این استاندارد ملی، باید به وضوح چگونگی ارائه‌ی این ویژگی را فراهم کند.

۳-۱-۱۳

وابسته به پیاده‌سازی[2]

ویژگی وابسته به پیاده‌سازی، ویژگی است که باید از سوی پیاده‌سازی مدعی انطباق با این استاندارد ملی ارائه شود اما پیاده‌سازی یاد شده نیازی به تعیین چگونگی ارائه‌ی ویژگی ندارد.

۳-۱-۱۴

پارامتر ورودی

یک پارامتر صوری است که پارامتر واقعی متناظر با آن باید از سمت رویه کارخواه برای رویه کارساز هنگام فراخوانی‌اش فراهم شود.

۳-۱-۱۵

پارامتر ورودی/خروجی

یک پارامتر صوری است که پارامتر واقعی متناظر با آن باید از سمت رویه کارخواه برای رویه کارساز هنگام فراخوانی‌اش و برای رویه کارخواه در بازگشت از رویه کارساز فراهم شود.

۳-۱-۱۶

بستار واسط[3]

مجموعه‌ای از نام‌ها و مجموعه‌ای از بستارهای رویه با نگاشتی بین آن‌ها است.

۳-۱-۱۷

زمینه اجرای واسط[4]

اجتماع زمینه‌های[5] اجرای رویه برای بستار واسط مشخص است.

1 - Implementation defined
2 - Implementation dependent
3 - Interface closure
4 - Interface execution context
5 - Context

۳-۱-۱۸

مرجع واسط[1]

شناسه‌ای که نمونه‌ای از یک واسط خاص را نشان می‌دهد.

۳-۱-۱۹

نوع واسط[2]

مجموعه‌ای از اسامی و مجموعه‌ای از انواع رویه با نگاشت بین آن‌ها است.

۳-۱-۲۰

شناسه‌ی نوع واسط[3]

شناسه‌ای است که نوع واسط را نشان می‌دهد.

۳-۱-۲۱

همبستگی احضار[4]

همبستگی احضارِ بستار رویه <تصویر، همبستگی> اعمال شده به مجموعه‌ای از مقادیر پارامتر واقعی، همبستگی مربوط به بستاری است که نگاشت تمامی نمادهای محلی به مقادیر و تمامی نمادهای پارامتر صوری به مقادیر پارامتر واقعی متناظرشان، به آن اضافه شده باشد. بنابراین همبستگی احضار، انقیادی به مقادیر تمام نمادها در تصویر رویه در طول احضار است.

۳-۱-۲۲

زمینه‌ی احضار[5]

در یک فراخوانی رویه خاص، نمونه اشیاء ارجاع شده به وسیله رویه که در آن عمر اشیاء با عمر فراخوانی محدود می‌شود را زمینه‌ی فراخوانی می‌گویند.

۳-۱-۲۳

مرتب کردن[6]

فرآیند جمع‌آوری پارامترهای واقعی، به احتمال تبدیل کردن آن‌ها، و هم‌گذاری آن‌ها برای انتقال است.

۳-۱-۲۴

پارامتر خروجی

پارامتر صوری که پارامتر واقعی متناظر با آن باید در بازگشت از رویه کارساز برای رویه کارخواه ارائه شود.

---

1 - Interface reference
2 - Interface type
3 - Interface type identifier
4 - Invocation association
5 - Invocation context
6 - Marshaling

۲۵-۱-۳

پارامتر

پارامتر برای انتقال مقداری از رویه کارخواه به کارساز استفاده می‌شود. مقداری که به وسیله رویه کارخواه فراهم می‌شود پارامتر واقعی است، پارامتر صوری برای تشخیص مقدار دریافتی در رویه کارساز به کار می‌رود.

۲۶-۱-۳

بستار رویه ناقص[1]

بستار رویه‌ی است که برخی از نمادهای سراسری آن نگاشته نمی‌شود. بستارهای رویه ممکن است کامل باشند که در آن تمام نمادهای سراسری نگاشته شده‌اند. اما در بستار رویه ناقص یکی یا بیشتر از یکی از نمادهای سراسری نگاشته نشده است.

۲۷-۱-۳

رویه

مقدار رویه است.

۲۸-۱-۳

فراخوانی رویه[2]

عمل احضار[3] یک رویه است.

۲۹-۱-۳

بستار رویه[4]

زوج <تصویر رویه، همبستگی> که در آن همبستگی، نگاشت نمادهای سراسری (نه مابقی نمادها) تصویر را تعریف می‌کند.

یادآوری - بستارهای رویه، مقادیر نوع رویه هستند که در انواع داده‌ای مستقل از زبان (ISO/IEC 11404) به آن‌ها اشاره شده است.

۳۰-۱-۳

زمینه‌ی اجرای رویه[5]

برای یک رویه خاص، نمونه‌ای از اشیاءای است که لازمه‌ی ارجاع خارجی برای اجرای رویه را برآورده می‌کنند. این اشیاء عمری طولانی‌تر از فراخوانی مجزای آن رویه دارند.

1 - Partial procedure closure
2 - Procedure call
3 - Invocation
4 - Procedure closure
5 - Procedure execution context

۳-۱-۳۱

تصویر رویه[1]

نمایش مقدار یک نوع رویه‌ی خاص است، که دنباله‌ی خاصی از دستوراتی که در هنگام فراخوانی رویه اجرا می‌شوند را در بر می‌گیرد.

۳-۱-۳۲

احضار رویه[2]

شئ‌ای است که سه‌تایی: تصویر رویه، زمینه‌ی اجرا و زمینه‌ی احضار را نشان می‌دهد.

۳-۱-۳۳

نام رویه

نام یک رویه در یک تعریف نوع واسط است.

۳-۱-۳۴

بازگشت رویه

عمل بازگشت از رویه کارساز با یک پایان‌دهی مشخص است.

۳-۱-۳۵

نوع رویه[3]

خانواده‌ای از انواع داده‌ای است که هریک از اعضای آن مجموعه‌ای از عملیات بر روی مقادیر انواع داده‌ای دیگر هستند. توجه داشته باشید که این تعریف با تعریف مقدار رویه متفاوت است.

۳-۱-۳۶

مقدار رویه[4]

دنباله‌ی بسته‌ای از دستورالعمل‌هایی است که از یک منبع خارجی به آن وارد شده و در خاتمه کنترل را به منبع خارجی باز می‌گرداند.

۳-۱-۳۷

پردازنده[5]

مترجم یا مفسری که در ترکیب با یک پیکربندی کار می‌کند.

1 - Procedure image
2 - Procedure invocation
3 - Procedure type
4 - Procedure value
5 - Processor

۳۸-۱-۳

رویه کارساز[1]

رویه‌ی است که به وسیله یک فراخوانی رویه احضار می‌شود.

۳۹-۱-۳

نماد[2]

هستاری[3] از برنامه است که برای ارجاع به یک مقدار به کار گرفته می‌شود.

۴۰-۱-۳

پایان‌دهی[4]

وضعیت از پیش تعریف شده مربوط به اتمام یک فراخوانی رویه است.

۴۱-۱-۳

نامرتب کردن[5]

فرآیند واهم‌گذاری[6] پارامترهای منتقل شده، به احتمال تبدیل کردن آن‌ها، برای استفاده‌ی رویه کارساز هنگام احضارش یا رویه کارخواه هنگام بازگشت از رویه است.

۴۲-۱-۳

مقدار[7]

مقدار مجموعه شامل تمام مقادیری است که امکان دارد در طول اجرای برنامه رخ دهند.

۲-۳ کوته‌نوشت‌ها

نشانه‌گذاری نحو انتزاعی-یک	Abstract Syntax Notation - One	ASN.1
نشانه‌گذاری تعریف واسط	Interface Definition Notation	IDN
انواع داده‌ی مستقل از زبان[8]	Language-Independent Datatypes	LID
فراخوانی رویه مستقل از زبان	Language-Independent Procedure Calling	LIPC

۴ انطباق[9]

یک پردازنده زبان می‌تواند با نگاشت سازوکار فراخوانی رویه ذاتی خود به مدل LIPC که در این استاندارد ملی تعریف می‌شود، مطابق با این استاندارد ملی شود.

1- Server procedure
2 - Symbol
3 - Entity
4 - Termination
5 - Unmarshaling
6 - Disassembling
7 - Value

۸ - که در ISO/IEC 11404:1995 تعریف شده‌اند.

9 - Conformance

یادآوری – واژه‌ی «پردازنده زبان» که در این بند استفاده می‌شود، می‌تواند بسط داده شود تا هر چیزی که اطلاعات را پردازش می‌کند و دارای سازوکاری برای یک فراخوانی رویه است را در بر بگیرد.

۴-۱ طریقه‌هاي انطباق

پردازنده زبانی که مدعی است مطابق این استاندارد ملی است باید با یک یا هر دو روشی که در ادامه بیان می‌شود، انطباق داشته باشد.

۴-۱-۱ انطباق طریقه کارخواه

یک پردازنده زبان جهت انطباق در طریقه کارخواه، باید به برنامه‌های نوشته شده به زبان خودش این امکان را بدهد که بتوانند با استفاده از فراخوانی رویه مستقل از زبان (LIPC) (که در بندهای ۵ و ۶ و ۷ این استاندارد ملی آورده شده است) رویه‌های نوشته شده به زبان دیگر (که پردازنده دیگری از آن‌ها پشتیبانی می‌کند) را فراخونی کنند. در این صورت گفته می‌شود که انطباق (و توانایی اجرا) در طریقه کارخواه وجود دارد. برای این منظور پردازنده زبان باید نگاشتی از مدل فراخوانی رویه خود به مدل LIPC تعریف کند.

یادآوری – اگر قرار باشد برنامه‌ای که از LIPC استفاده می‌کند، بین پردازنده‌هایی که مطابق با طریقه کارخواه هستند قابلیت جابه‌جایی داشته باشد، برنامه و پردازنده‌ها باید از استاندارد زبان مربوطه و از استانداردهایی که آن زبان را مقید به استانداردهای LIPC و LID می‌کنند نیز پیروی کنند.

۴-۱-۲ انطباق طریقه کارساز

یک پردازنده زبان جهت انطباق در طریقه کارساز، باید به برنامه‌های نوشته شده در زبان دیگر اجازه بدهد با استفاده از فراخوانی رویه مستقل از زبانی (LIPC) که در بندهای ۵ و ۶ و ۷ این استاندارد ملی آورده شده است، رویه‌های نوشته شده به زبان خودش را فراخوانی کند (یعنی پردازنده، فراخوانی‌های رویه تولید شده به وسیله پردازنده طریقه کارخواهی که برنامه‌ای به زبان دیگر را اجرا می‌کند را قبول و اجرا خواهد کرد و پس از اتمام کار، کنترل را به پردازنده طریقه کارخواه باز می‌گرداند). در این صورت گفته می‌شود که انطباق (و توانایی اجرا) در طریقه کارساز وجود دارد. برای این منظور پردازنده زبان باید نگاشتی از مدل LIPC به مدل فراخوانی رویه خودش تعریف کند.

یادآوری ۱- پردازنده کارخواه می‌تواند از مدل فراخوانی رویه تعریف شده در این استاندارد ملی، برای فراخوانی رویه‌های با زبان یکسان که روی پردازنده کارساز اجرا می‌شوند نیز استفاده کند. همچنین در صورتی که پردازنده‌ای هم منطبق بر طریقه کارخواه و هم منطبق بر طریقه کارساز باشد، با استفاده از این مدل می‌تواند حتی به خودش هم خدمت کند.

یادآوری ۲- در صورتی که قرار است رویه‌ی بین پردازنده‌هایی که مطابق با طریقه کارساز هستند، جابه‌جا شود و کماکان به وسیله برنامه‌ها و پردازنده‌های کارخواه فراخوانی شود، رویه و پردازنده‌ها باید مطابق با استاندارد زبان مربوطه و استانداردهایی که آن زبان را مقید به استانداردهای LIPC و LID می‌کنند نیز باشند.

۵ مدل فراخواني رويه: توصيف غيرصوري

۵-۱ مروري بر مدل

یک رویه به صورت دنباله‌ی بسته‌ای تعریف می‌شود که از یک منبع بیرونی به آن وارد شده و کنترل را به آن منبع خارجی برمی‌گرداند.
ساختار کلی یک فراخوانی رویه را می‌توان با اجرای تک نخی[1] برنامه‌ی خاصی که در آن کنترل جریان از رویه‌ای به رویه دیگر منتقل می‌شود، شرح داد. آغازکننده‌ی فراخوانی، رویه کارخواه و رویه‌ی که فراخوانی می‌شود را رویه کارساز می‌نامند.

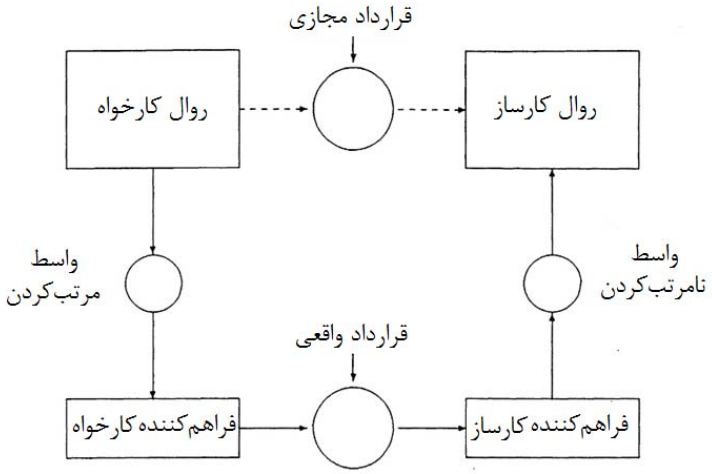
یادآوری ۱- یک رویه کارساز در صورتی که برای تکمیل عمل مورد نظر خود رویه دیگری را فراخوانی کند، می‌تواند رویه کارخواه نیز باشد.

رویه‌ها می‌توانند داده را بین کارخواه و کارساز از طریق پارامترها مبادله کنند (به بند ۵-۲ مراجعه شود). به علاوه رویه‌های کارخواه و کارساز می‌توانند با استفاده از داده‌های سراسری، داده‌ها را به اشتراک بگذارند (به بند ۵-۲-۲ مراجعه شود). برای این‌که پارامترهایی که به وسیله رویه کارخواه مشخص می‌شوند به درستی تفسیر شوند باید برای انتقال، آن‌ها را به شکل پایه‌ای که برای هر دوی رویه کارخواه و کارساز مشترک است، تبدیل کرد (عمل مرتب کردن) (به بند ۵-۲-۳ مراجعه شود). پس از انتقال داده‌ها، رویه کارساز باید داده‌ها را از شکل پایه به انواع داده‌ای تعریف شده در زبان کارساز یا زبانی که آن زبان را مقید به انواع داده‌ای مستقل از زبان (ISO/IEC 11404) می‌کند، تبدیل کند (عمل نامرتب کردن) (به بند ۵-۲-۳ مراجعه شود).

یادآوری ۲- مثالی از فرآیند مرتب و نامرتب کردن پارامترها، زمانی است که رویه کارخواه نوشته شده به زبان پاسکال[2]، رویه کارسازی در زبان فورترن[3] را با ارسال مقدار یک پارامتر از نوع نویسه[4] فراخوانی می‌کند. نوع داده‌ی «char» در پاسکال به یک نویسه‌ی LID نگاشته خواهد شد. برای این‌که نویسه‌ی LID بتواند به رویه کارساز منتقل شود، باید به مقدار مناسبی در ASN.1 (شکلی که هم به وسیله رویه کارخواه و هم به وسیله رویه کارساز فهمیده می‌شود) مرتب شود. سپس مقدار ASN.1 به کارساز منتقل خواهد شد و در هنگام دریافت در سمت کارساز، به یک نویسه‌ی LID نامرتب می‌شود؛ و در ادامه این نویسه‌ی LID به یک «character*1» در زبان فورترن نگاشته می‌شود.

نمودار زیر مؤلفه‌های اصلی مدل فراخوانی مستقل از زبان را نشان می‌دهد:

1 - Single thread
2 - Pascal
3 - Fortran
4 - Character



شکل ۱ - مدل فراخوانی رویه مستقل از زبان

این مدل، نحوه‌ی ارتباط رویه‌های کارساز و کارخواه را نشان می‌دهد که پیاده‌سازی آن‌ها مطابق این استاندارد ملی است. قرارداد مجازی بین رویه کارخواه و رویه کارساز به وسیله نشانه‌گذاری تعریف واسط (IDN) این استاندارد ملی تعریف می‌شود. هنگام ایجاد نمونه[1] از یک فراخوانی، واسط مرتب کردن، پارامترها را مرتب می‌کند و این اطلاعات را به متصدی[2] LIPC کارخواه می‌فرستد. متصدی LIPC کارخواه از طریق قرارداد واقعی که قابل ارسال (برای مثال ASN.1) است، به متصدی کارساز متصل است. سپس متصدی LIPC کارساز داده‌ها را به وسیله واسط نامرتب کردن به شکلی که برای رویه کارساز قابل فهم باشد، نامرتب می‌کند. در زمان بازگشت، فرآیند مذکور به صورت معکوس انجام می‌گیرد (واسط نامرتب کردن به واسط مرتب کردن و واسط مرتب کردن به واسط نامرتب کردن، تغییر می‌کند).

۲-۵ ارسال پارامتر[3]

هر نوع داده تعریف شده در انواع داده‌ای مستقل از زبان - ISO/IEC 11404 می‌تواند نوع داده‌ی یک پارامتر صوری مربوط به فراخوانی رویه مستقل از زبان باشد. این استاندارد ملی ارسال پارامتر را فقط برای ارسال مقادیر تعریف می‌کند. بنابر این، یک پارامتر واقعی، هر مقداری از نوع داده‌ای است که در فراخوانی نیاز است. مدل ارسال پارامتر تعریف شده در این استاندارد، مدل به‌طور قوی نوع‌دار[4] است.

یادآوری ۱- زبانی که مبتنی بر این استاندارد است می‌تواند نوع‌دهی ضعیف[5] را با راحت‌تر گرفتن قواعد همبستگی و افزودن تبدیل‌های نوع داده به صورت ضمنی داشته باشد.

یادآوری‌های زیر، سازوکارهای ارسال پارامتر رایجی که در زبان‌های کنونی وجود دارند را به چهار روش ارسال پارامتری که در این استاندراد تعریف شده است، ربط می‌دهند.

1 - Instantiation
2 - Provider
3 - Parameter passing
4 - Strongly typed
5 - Weak typing

یادآوری ۲- فراخوانی با مقدار (پارامترهای ورودی): فراخوانی با مقدار، ساده‌ترین نوع ارسال پارامتر در بین تمام سازوکارهای رایج ارسال پارامتر است و به‌طور مستقیم در LIPC به وسیله فراخوانی با مقدار ارسال شده در زمان راه‌اندازی (به بند ۵-۲-۱-۱ مراجعه شود) پشتیبانی می‌شود. قرارداد مجازی به وسیله ارزیابی پارامتر واقعی در کارخواه و ارسال مقدار آن به رویه کارساز، و دریافت آن در رویه کارساز انجام می‌گیرد. نیازی نیست رویه کارخواه کار اضافه دیگری انجام دهد. رویه کارساز، کاری را که می‌خواهد با مقدار دریافت شده انجام می‌دهد، اما نمی‌تواند درخواست دیگری از کارخواه در ارتباط با پارامتر واقعی که مقدار را تولید کرده است داشته باشد.

یادآوری ۳- فراخوانی با بازگشت مقدار (پارامترهای خروجی): این سازوکار رایج برای ارسال پارامتر نیز به‌طور مستقیم در LIPC به وسیله فراخوانی با مقدار بازگشتی مشخص شده، پشتیبانی می‌شود. قرارداد مجازی این سازوکار شامل مفهوم ارسال یک پارامتر فقط به عنوان ابزاری جهت دریافت یک مقدار است. در صورتی که در قیود یک زبان مشخص، پارامتری در سطح پردازنده زبان ارسال شود، چیزی که ارسال می‌شود اشاره‌گر ضمنی به مقدار نوع داده‌ای مورد نظر است که رویه کارساز آن را تنظیم می‌کند. رویه کارساز نمی‌تواند به مقدار نوع داده قبل از فراخوانی دسترسی داشته باشد. برخی زبان‌ها در مدل نوع داده‌ای‌شان به صورت صریح بین انواع داده‌ای مقدارهای نگه داشته شده به وسیله متغیرها و انواع داده‌ای خود متغیرها تفاوت قائل می‌شوند. به عنوان مثال برخی از زبان‌ها یک عملگر محتوای مرجع[1] صریح دارند (که مقدار مربوطه را باز می‌گرداند). مدل LIPC به زبان‌های فاقد چنین مدلی اجازه می‌دهد چنین تفاوتی را در سطح قرارداد خدمت زبان، بدون به هم ریختن مدل قرارداد مجازی داشته باشند.

یادآوری ۴- فراخوانی با ارسال و بازگشت مقدار (پارامترهای ورودی-خروجی): این سازوکار ارسال پارامتر رایج، یک سازوکار ورودی/خروجی است که در آن پارامتر واقعی می‌تواند به عنوان مقصدِ فراخوانی با بازگشت مقدار در پایان‌دهی (به بند ۵-۲-۱-۳ مراجعه شود) ارزیابی شود. اگرچه در مدل LIPC به عنوان پارامتری هم با آن خصوصیت و هم خصوصیت فراخوانی با مقدار ارسال شده در زمان راه‌اندازی (به بند ۵-۲-۱-۱ مراجعه شود) در نظر گرفته می‌شود. به طور معادل این پارامتر می‌تواند به دو پارامتر ضمنی از این دو نوع بسط داده شود. پارامتر واقعی متناظر با یک پارامتر صوری از نوع داده‌ی «t»، باید بتواند هنگام ارزیابی، مقصدی برای چنین مقداری به دست آورد (یعنی اشاره‌گری صریح یا ضمنی به مقداری از نوع داده‌ی «t»). برای قسمت «ورودی» مشخصات ورودی/خروجی، مقدار نگه داشته شده در آن مقصد در هنگام آغاز فراخوانی، به وسیله رویه کارخواه بازیابی شده و به رویه کارساز رله می‌شود. خود مقصد نیز ثبت می‌شود. در قرارداد مجازی، کارخواه مقدار بازگشتی (قسمت «خروجی» مشخصات ورودی/خروجی) را از رویه کارساز دریافت می‌کند و آن را به مقصد ثبت شده می‌فرستد.

زمانی که قیود زبان یا قرارداد خدمت، خود مقصد را به عنوان رونوشت ورودی/رونوشت خروجی[2] به رویه کارساز می‌فرستد، رویه کارساز باید برای بازیابی مقدار «ورودی» به صورت آنی در زمان انتقال اقدام کند و مقدار «خروجی» بازگشتی را در پایان فراخوانی به مقصد باز گرداند. در طول فراخوانی، کارخواه به صورت ضمنی یا صریح مقصد را برای رویه کارساز با «فقط یکبار خواندنی، فقط یکبار نوشتنی»[3] علامت می‌گذارد و هر تلاشی به وسیله رویه کارساز برای نقض آن شرط، خطا محسوب می‌شود.

یادآوری ۵- فراخوانی با مرجع: در این مورد یک پارامتر صوری از نوع داده‌ی «t» اشاره‌گر ضمنی به «t» است و پارامتر واقعی باید به‌طور متناظر به چنین اشاره‌گری ارزیابی شود. سپس اشاره‌گر به «t» به عنوان پارامتر «ورودی» با مقدار فرستاده می‌شود.

1 - Dereference operator
2 - Copy-in/copy-out
3 - Read once only, write once only

اشاره‌گر به عنوان پارامتر ورودی/خروجی ارسال نمی‌شود زیرا سطح اضافی‌تری از آدرس‌دهی غیرمستقیم را موجب می‌شود.

قرارداد مجازی به این صورت است که کارخواه مسیر دسترسی به مقصد را ارائه می‌کند. مقصد ثابت است، اما مسیر دستیابی می‌تواند به وسیله رویه کارساز هم برای خواندن و هم برای نوشتن مقادیری از نوع داده‌ی «t» استفاده شود. در ارتباط-محکم[1]، قرارداد خدمت می‌تواند شامل ارسال مقصد واقعی باشد و نیازی نیست کارخواه تا پایان فراخوانی عملی را انجام دهد. در محیط خدمت ارتباط-ضعیف[2]، قرارداد خدمت کارخواه در طول فراخوانی عمل کرده و به درخواست‌های کارساز برای خواندن یا نوشتن نوع داده‌ی «t» پاسخ می‌دهد. در عمل این فراخوانی یک فراخوانی دو طرفه است که جهات «ورودی» و «خروجی» آن معکوس شده‌اند.

این فراخوانی‌های دوطرفه در فراخوانی با مرجع در یک محیط ارتباط-ضعیف، سربار زیادی دارند که ممکن است منجر به عدم پشتیبانی فراخوانی با مرجع در چنین خدماتی شود.

۵-۲-۱ روش‌های ارسال پارامتر

چهار روش ارسال پارامتر پایه در این استاندارد تعریف شده است:

۱- فراخوانی با مقدار ارسال شده در زمان راه‌اندازی[3]

۲- فراخوانی با مقدار ارسال شده در زمان تقاضا[4]

۳- فراخوانی با مقدار بازگشتی در زمان پایان‌دهی[5]

۴- فراخوانی با مقدار بازگشتی هنگام در دسترس بودن[6]

۵-۲-۱-۱ فراخوانی با مقدار ارسال شده در زمان راه‌اندازی

این فراخوانی، ساده‌ترین شکل ارسال پارامتر است. پارامتر صوری رویه کارساز مقداری از نوع داده‌ی مربوطه‌اش را دریافت می‌کند. قرارداد مجازی به این صورت است که کارخواه، پارامتر واقعی را ارزیابی می‌کند و نتیجه را هنگام انتقال کنترل ارسال می‌کند. رویه کارساز این مقدار را دریافت می‌کند و هیچ تعامل دیگری در ارتباط با این پارامتر اتفاق نمی‌افتد.

یادآوری – در اصطلاح رایج به این شکل از ارسال پارامتر، فراخوانی با مقدار می‌گویند.

۵-۲-۱-۲ فراخوانی با مقدار ارسال شده در زمان تقاضا

قرارداد مجازی در این نوع از ارسال پارامتر به این صورت است که کارخواه فقط زمانی که درخواستی از رویه کارساز داشته باشد، پارامتر واقعی را ارزیابی می‌کند و مقدار نتیجه را ارائه می‌کند. ارزیابی و ارسال پارامتر واقعی فقط و فقط زمانی انجام می‌گیرد که رویه کارساز آن را درخواست کرده باشد. این درخواست می‌تواند در آغاز فراخوانی یا در طول فراخوانی باشد.

تفاوت اصلی این روش با فراخوانی با مقدار ارسال شده در زمان راه‌اندازی این است که مقدار ارسال شده در برخی موارد متفاوت خواهد بود.

---

1 - Close-coupled
2 - Loosely-coupled
3 - Call by Value Sent on Initiation
4 - Call by Value Sent on Request
5 - Call by Value Returned on Termination
6 - Call by Value Returned when Available

یادآوری ۱- اگرچه این سازوکار به عنوان لازمه‌ی یک استاندارد صریح در زبان‌های برنامه‌نویسی رایج نیست، اما یک سازوکار بهینه‌سازی برای پیاده‌سازی زبان‌های برنامه‌نویسی است.

یادآوری ۲- مثالی از استفاده از فراخوانی با مقدار ارسال شده در زمان تقاضا، زمانی است که کارخواه از رویه کارساز می‌خواهد زمانی را در نقطه‌ی مشخصی در طول اجرای آن فراخوانی (نه در زمان راه‌اندازی فراخوانی) ثبت کند.

یادآوری ۳- کارساز می‌تواند از پارامتری که نوع فراخوانی با مقدار ارسال شده در زمان تقاضا دارد برای فراخوانی یک پارامتر رویه‌ی ضمنی که در آن رویه کارساز ارزیابی را یک مرتبه انجام می‌دهد، استفاده کند؛ در این صورت ارجاع‌های دیگر رویه کارساز به آن پارامتر صوری از همان مقدار استفاده می‌کند و رویه کارساز درخواست دیگری برای مقدار نخواهد داشت.

۵-۲-۱-۳ فراخوانی با مقدار بازگشتی در زمان پایان‌دهی

قرارداد مجازی در این شکل از ارسال پارامتر به این صورت است که در پایان فراخوانی، رویه کارساز مقداری از نوع داده‌ی پارامتر صوری تهیه کرده و کارخواه پس از دریافت مقدار بازگشتی آن را به مقصد مناسب می‌فرستد.

یادآوری ۱- این شکل از ارسال پارامتر تحت عنوان فراخوانی با بازگشت مقدار شناخته می‌شود. و در واقع معادل «خروجی» فراخوانی با مقدار ارسال شده در زمان راه‌اندازی است.

به طور مفهومی، این رویه کارخواه است که مقدار بازگشتی را به مقصد می‌فرستد و نه رویه کارساز، زیرا زبان یا نگاشت کارخواه تفسیر مقصد و فرآیند بازگشت را تعیین می‌کند.

یادآوری ۲- در یک محیط با ارتباط-محکم، که در آن ارائه‌ی مقصد واقعی (حتی شاید آدرس سخت‌افزاری) به رویه کارساز، کار جزئی است، دلیلی ندارد که قرارداد خدمت واقعی در سطح پیاده‌سازی نباید شامل ارائه‌ی مقصد واقعی برای رویه کارساز شود که در این صورت مقدار بازگشتی خود را به‌طور مستقیم به آن مقصد می‌فرستد. این مورد، کارکرد اضافی‌تری در سطح خدمت است که رویه کارساز برای رویه کارخواه انجام می‌دهد و تأثیری بر تقسیم‌بندی منطقی مسئولیت‌ها در سطح قرارداد مجازی ندارد.

یادآوری ۳- در مورد رویه‌های تابعی[1] این نوع از ارسال پارامتر با بازگشت مقدار برای رویه به صورت کلی همخوانی دارد. ارسال پارامتری که فراخوانی با مقدار بازگشتی در پایان‌دهی را به کار می‌گیرد، با رویه‌های تابعی از طریق استفاده از یک پارامتر بی‌نام[2] اضافی، سازگار می‌شوند.

۵-۲-۱-۴ فراخوانی با مقدار بازگشتی هنگام آماده بودن

در این شکل از ارسال پارامتر، رویه کارساز مقدار پارامتر را زمانی که مقدار بازگشتی آماده شود، بر می‌گرداند. این مقدار می‌تواند در طول اجرای فراخوانی، در پایان فراخوانی، یا زمان‌های بعدی بازگردانده شود. زمان انتخاب شده به وسیله انقیاد خدمت مبتنی بر LIPC تعریف می‌شود و به وسیله این استاندارد ملی تعریف نمی‌شود. مدل LIPC فقط نیاز دارد که این امکان در نظر گرفته شده باشد. قرارداد مجازی به این صورت است که هر زمان که رویه کارساز مقدار را باز می‌گرداند، کارخواه آن را می‌پذیرد و مقدار بازگشتی را به مقصد مناسب می‌فرستد.

1 - Function procedure
2 - Anonymous

یادآوری – در این نوع ارسال پارامتر، امکان این‌که مقدار بازگشتی بیش از یک بار بازگردد هم وجود دارد.

۵-۲-۲ داده‌های سراسری[1]

اصطلاح داده‌ی سراسری برای داده‌هایی استفاده می‌شود که در یک زمینه‌ی اجرایی مشترک تعریف می‌شوند و می‌توانند به وسیله رویه دیگری که در زمینه‌ی احضار متفاوتی در زمینه‌ی اجرایی یکسان در حال اجرا است، مورد ارجاع قرار گیرند. به طور مفهومی، داده‌ی سراسری نیاز دارد تا مرتب و نامرتب کردن داده‌ی سراسری در واحدهای اطلاعاتی مجزا انجام شود. پیاده‌سازی‌های منطبق بر این استاندارد ملی ممکن است از یک سازوکار تعریف شده در پیاده‌سازی جهت به اشتراک گذاردن داده‌ی سراسری پشتیبانی کنند. همچنین ممکن است از افراز[2] داده‌ی سراسری نیز پشتیبانی کنند. افراز داده به معنای قابلیت جدا کردن داده از یک رویه است. توصیه می‌شود که پیاده‌سازی‌ها، داده‌ی سراسری را به وسیله پارامترهای ضمنی که در هنگام فراخوانی ارسال می‌شوند، پشتیبانی کنند، اما این سازوکار تنها روش معتبر در مواردی که عملیات مرتب/نامرتب کردن عملی ناچیز است، نیست.

یادآوری ۱– در IDN، داده‌ی سراسری به عنوان یک پارامتر صریح به رویه نمایش داده می‌شود. در یک نگاشت زبان خاص، می‌توان این پارامترهای صریح را با استفاده از سازوکارهایی مانند متغیرهای خارجی و پارامترهای ضمنی فراهم کرد.

یادآوری ۲– داده‌ی سراسری زمانی که کارساز آن را احتیاج دارد باید در دسترس باشد (یعنی قبل از احضار، در زمان احضار، قبل از این‌که استفاده از آن نیاز باشد، یا زمانی که دستیابی به آن نیاز است).

یادآوری ۳– سازوکاری که به وسیله آن اشیاء در زمینه احضار به اشیاء سراسری همبسته می‌شوند، ممکن است به وسیله زبان یا نگاشت زبان تعریف شود و یا این‌که به پیاده‌سازی واگذار شود.

۵-۲-۳ مرتب/ نامرتب کردن پارامترها

داده‌ای که بین رویه کارخواه و کارساز رد و بدل می‌شود، باید به شکل قابل انتقالی هم‌گذاری شود. این شکل قابل انتقال به رویه‌های کارخواه و کارساز امکان می‌دهد تا داده‌های نگاشته شده به LID خود را به شکلی کدبندی کنند که هم برای فراخوانی مستقل از زبان در همان سامانه و هم فراخوانی رویه راه دور مناسب باشد. مشخصات این شکل قابل انتقال خارج از حوزه‌ی این استاندارد ملی است.

یادآوری – نشانه‌گذاری نحو انتزاعی- یک، مشخصات مناسب یک شکل قابل انتقال است.

مرتب کردن داده عملی است که رویه کارخواه باید جهت تغییر شکل داده‌اش به شکل قابل انتقال برای رویه کارساز انجام دهد. نامرتب کردن داده عملی است که رویه کارساز باید برای گرفتن داده‌ی فرستاده شده به وسیله رویه کارخواه و تبدیل آن به داده‌ی مناسب در زبان رویه کارساز انجام دهد. مرتب کردن محدود به فراخوانی رویه نیست. در زمان بازگشت، رویه کارساز باید همه‌ی داده‌های بازگشتی را به شکلی که در هر دو رویه مشترک است، مرتب کند. نامرتب کردن داده محدود به رویه کارساز نیست، زیرا رویه کارخواه باید بتواند هر داده‌ای که به وسیله رویه کارساز باز گردانده می‌شود را نامرتب کند.

---

1 - Global data
2 - Partitioning

از آن‌جایی که مرتب و نامرتب کردن داده‌ها در فراخوانی‌های رویه به‌طور معمول پیچیده هستند و باعث افت کارایی می‌شوند، یک پیاده‌سازی ممکن است این فرآیند را هر جایی که ممکن باشد، بهینه کند. به احتمال بهینه‌سازی، زمانی است که سامانه‌های کارخواه و کارساز یکسان هستند و زبان‌هایی که در فراخوانی رویه دخیل هستند نمایش داده‌ی یکسانی دارند.

۵-۲-۴ پارامترهای اشاره‌گر

فراخوانی با مقدار ارسال شده در زمان راه‌اندازی یک اشاره‌گر، امکان دسترسی به هستاری که به آن اشاره شده است را می‌دهد. پس از فراخوانی، رویه کارساز نمی‌تواند مقدار اشاره‌گر را برای این‌که به چیز دیگری ارجاع کند، تغییر دهد.

یادآوری - برای مثال در صورتی که مقدار ارسال شده اشاره‌گری به یک رکورد[1] باشد، پس از فراخوانی هنوز هم اشاره‌گر به همان رکورد اشاره می‌کند، حتی اگر مقادیر فیلدهای آن رکورد تغییر کرده باشند.

در صورتی که تغییر چیزی که اشاره‌گر به آن ارجاع می‌کند ضروری باشد، باید سطح دیگری از ارجاع غیرمستقیم را چه به صورت مستقیم (به‌عنوان مثال به وسیله فراخوانی با مرجع) و چه به صورت غیر مستقیم (به‌عنوان مثال به وسیله فراخوانی با مقدار بازگشتی) به کار گرفت. مسیر دسترسی از طریق پارامترهای اشاره‌گر، بر دسترسی به تمامی سطوح پایین‌تر (که حاوی مقادیر انواع داده‌ی اصلی[2] هستند که به وسیله اشاره‌گرهای با پایین‌ترین سطح به آن‌ها ارجاع می‌شود) دلالت دارد.

۵-۲-۵ انواع خصوصی[3]

یک نوع خصوصی، نوع داده‌ای است که از تغییرات داخل رویه کارساز (صرف‌نظر از صفات پارامتری که با نوع خصوصی ارسال می‌شود) محافظت می‌شود. نباید هیچ‌گونه عملیاتی روی پارامتری با نوع خصوصی مجاز باشد. نوع خصوصی با کلمه‌ی کلیدی «restricted» قبل از نوع داده‌ای LID در IDN تعریف می‌شود.

یادآوری - یک نوع خصوصی می‌تواند به عنوان جریان هشت‌تایی[4] در نظر گرفته شود که عملیاتی بر روی آن انجام نشده است.

۵-۳ کنترل زمان اجرا[5]

۵-۳-۱ پایان‌دهی‌ها[6]

پیاده‌سازی که منطبق بر این استاندارد ملی است، باید روشی برای وقوع و مدیریت پایان‌دهی‌هایی که در طول راه‌اندازی، اجرا، یا اتمام فراخوانی رویه رخ می‌دهند، فراهم کند. وقوع یک پایان‌دهی در صورت لزوم دلالت بر این ندارد که رویه کارساز باید در همان لحظه پایان یابد؛ اما پایان‌دهی‌هایی که ایجاد می‌شوند نباید در پیاده‌سازی نادیده گرفته شوند. مثال‌هایی از پایان‌دهی‌های ممکن به شرح زیر است:

- پایان‌دهی‌عادی احضار رویه‌ای که پارامترهای خروجی، پارامترهای ورودی/خروجی، و نتیجه را (در صورت وجود) بر می‌گرداند.

1 - Record
2 - Primitive datatype
3 - Private types
4 - Octet stream
5 - Execution-time Control
6 - Terminations

- پایان‌دهی غیرعادی، که در آن خود رویه خطا یا شرایط غیرمعمولی را تشخیص می‌دهد.
- لغو از بیرون، که در آن هستار دیگری تعیین می‌کند که رویه باید پایان یابد.
- رویدادهای[1] سخت‌افزاری یا نرم‌افزاری تشخیص داده شده‌ای که ممکن است برای اجرای درست برنامه کاربردی حیاتی باشند یا نباشند.
- رویدادها یا اعلان‌های ناهمگام[2]
- عدم تطابق مقدار یا نوع در ارسال یا بازگشت پارامتر
- خرابی خود خدمت احضار شده‌ی اصلی

۵-۳-۱-۱ پایان‌دهی عادی

رویه‌ی که به صورت عادی پایان می‌یابد، پایان‌دهی ایجاد می‌کند که نشان‌دهنده‌ی بازگشت عادی است. رویه ممکن است پایان‌دهی‌های دیگری را هم گزارش کند. به عنوان مثال، در بازگشت از یک فراخوانی رویه همگام[3]، رویه ممکن است دو یا بیشتر از دو پایان‌دهی را بازگرداند. اما اولین این پایان‌دهی‌ها باید مشخص کند که پایان‌دهی عادی، غیرعادی، یا از طریق یک لغو است. در صورتی که فراخوانی رویه همگام باشد، رویه ممکن است قبل از، در طول یا پس از پایان‌دهی، یک کد پایان‌دهی اضافی نیز برگرداند.

۵-۳-۱-۲ پایان‌دهی غیرعادی

رویه‌ی که به صورت غیرعادی پایان می‌یابد، پایان‌دهی ایجاد می‌کند که نتیجه‌ی شرایطی غیر از دستور لغو خارجی است. یک رویه به‌طور معمول در صورتی که با شرایطی مواجه شود که ادامه‌ی اجرای آن را غیر ممکن می‌سازند، یا کامل کردن عملی (یا عمل‌هایی) که رویه کارخواه درخواست کرده است، به شکل دلخواه میسر نباشد، به صورت غیر عادی پایان می‌یابد. رویه کارخواه به وسیله سازوکار ایجاد پایان‌دهی تعریف شده در پیاده‌سازی از این مورد مطلع می‌شود. پایان‌دهی‌های غیرعادی می‌توانند به دو مورد زیر تقسیم‌بندی شوند:

- رویه به واسطه‌ی منطق خودش یک پایان‌دهی غیرعادی را تشخیص می‌دهد و به دنبال آن یک رویه توقف[4] صریح را اجرا می‌کند.
- در طول اجرای یک رویه، یک پایان‌دهی غیرعادی رخ می‌دهد که سبب می‌شود خطایی در سطوحی پایین‌تر از منطق رویه ایجاد شود. خطا باعث می‌شود کنترل به روال[5] مدیریت خطای رویه‌ی برود که رویه را مانند مورد قبل به پایان می‌رساند.

یک مورد خاص از پایان‌دهی غیرعادی هنگامی است که مقدار یکی یا بیشتر از یکی از پارامترهای واقعی در فراخوانی رویه اشتباه باشد. برای مثال، زمانی که مقدار پارامتر نوع داده‌ی اشتباهی دارد یا نوع داده‌ی درستی دارد اما خارج از محدوده‌ی مجاز برای آن است. در این‌جا می‌توان بین مقادیر پارامتری که شرایط واسط رویه‌ی که در IDN مشخص شده است را نقض می‌کنند و مقادیری (یا ترکیب مقادیری) که محدودیت‌های

1 - Event
2 - Asynchronous
3 - Synchronous
4 - Abort procedure
5 - Routine

مختص کاربرد[1] را (که نمی‌توانند در صورت‌بندی IDN مشخص شوند و بنا بر این باید به وسیله خود رویه به‌طور صریح بررسی شوند) نقض می‌کنند، تمایز قائل شد. اگرچه از دید رویه کارخواه، تنها تفاوت مابین این دو مورد این است که در مورد اول، خطا یکی از خطاهای مجموعه‌ی از پیش تعریف شده در این استاندراد ملی است(به بند ۵-۳-۱-۴ مراجعه شود). و مورد دوم یک کد مختص کاربرد است که در استاندارد دیگری (به احتمال مختص کاربرد) مشخص شده است.

۵-۳-۱-۳ لغو خارجی[2]

رویه در صورتی با لغو خارجی پایان پیدا می‌کند که دستوری که از خارج از رویه صادر شود، سبب پایان یافتن یا پایان داده شدن رویه شود. در مورد فراخوانی ناهمگام[3]، لغو ممکن از سمت رویه کارخواه باشد. هم در فراخوانی همگام و هم در فراخوانی ناهمگام، دستور لغو رویه می‌تواند از یک منبع خارجی بیاید (یعنی خارج از مدل LIPC). این دو مورد برای رویه کارساز غیر قابل تمایز هستند. در هر دو مورد، رویه کارخواه به وسیله سازوکار ایجاد پایان‌دهی که در پیاده‌سازی تعریف می‌شود، اعلانی را دریافت می‌کند.

۵-۳-۱-۴ شرایط از پیش تعریف‌شده

پیاده‌سازی که با این استاندارد ملی انطباق دارد، حداقل باید پایان‌دهی‌های زیر در طول فراخوانی رویه را گزارش کند:

- در دسترس نبودن رویه کارساز، عدم اجرای فراخوانی
- رویه کارخواه یا کارساز، نگاشت تعریف شده‌ای به IDN ندارد
- مقدار خارج از محدوده‌ی نوع داده‌ی پارامتر
- لغو کردن فراخوانی
- کافی نبودن منابع جهت کامل کردن فراخوانی
- اتمام عادی فراخوانی

۵-۴ کنترل اجرا

۵-۴-۱ فراخوانی همگام و ناهمگام

این‌که آیا یک فراخوانی همگام یا ناهمگام اجرا شود یا نشود، خارج از حیطه‌ی این استاندراد ملی است. مدل LIPC فراخوانی‌های همگام یا ناهمگام را منع نمی‌کند. یک پیاده‌سازی می‌تواند انتخاب کند که تعداد نخ‌های اجرایی در هر محیط فراخوانی خاص را محدود کند یا خیر.

۵-۴-۲ بازگشت[4]

مدل LIPC مانع استفاده از فراخوانی‌های بازگشتی نمی‌شود. این‌که چگونه یک پیاده‌سازی فراخوانی رویه بازگشتی را پیاده می‌کند، خارج از حیطه‌ی این استاندارد ملی است.

1 - Application-specific
2 - External cancellation
3 - Asynchronous
4 - Recursion

یادآوری - پیاده‌کننده باید مراقب باشد که ملاحظات بهینه‌سازی برای فراخوانی‌های LIPC، مسئله‌ی بازگشت را هم در نظر گرفته باشند.

۶ مدل فراخواني رويه: توصيف صوري

در این بند مدل رویه‌ها، متغیرها، قیود اسامی، محیط‌های اجرایی، و احضار ارائه می‌شود. یک سری از انواع داده‌ای جدید معرفی می‌شوند. برخی از این‌ها به‌طور مستقیم متناظر با مفاهیم برنامه‌نویسی هستند (مانند متغیرها)، و برخی دیگر نیز فقط برای کمک به تعاریف دیگر استفاده می‌شوند.

۶-۱ مقدار

مجموعه Value شامل تمامی مقادیری است که ممکن است در اجرای برنامه رخ دهند. مجموعه Value، تمام مقداری که با استفاده از انواع داده‌ای، مولدهای نوع، و سازوکارهای تعریفی انواع داده‌ای مستقل از زبان – ISO/IEC 11404 قابل تعریف هستند را در بر می‌گیرد. همچنین Value، شامل جعبه‌ها و بستارهای رویه (به بند ۶-۶ مراجعه شود) نیز می‌شود.

۶-۲ جعبه‌ها و حالت سراسري

جعبه اصطلاح اصلی برای دربردارنده‌ای است که مقداری با نوع‌داده‌ی ویژه‌ای را نگه می‌دارد، برای مثال آنچه که در برخی از زمینه‌ها «متغیر» نامیده می‌شود. جعبه‌ها در زمان اجرا، وجود دارند و فرابری می‌شوند[1] و در متن برخی از برنامه‌ها ممکن است به وسیله شناسه‌ها نام‌گذاری شوند. اما آن‌ها با چنین تصورات نحوی متفاوت هستند. جعبه‌ها دلالت بر هیچ‌گونه سازوکار پیاده‌سازی خاصی (مانند دخیره‌سازی) ندارند. سه عمل بر روی جعبه‌ها تعریف می‌شود:

create: → Value

write: Box * Value →

read: Box → Value

به سه خط بالا، *امضاء*[2] گفته می‌شود. هر امضاء اسم یک عمل، نوع ورودی‌های آن عمل (در صورت وجود)، و نوع خروجی‌های آن (در صورت وجود) را فهرست می‌کند. یک * (عملگر ضرب دکارتی) انواع ورودی (یا خروجی) را از هم جدا می‌کند.

عمل Create (ایجاد) یک جعبه‌ی جدید را به وجود می‌آورد. عمل Write (نوشتن)، مقدار جدیدی را به جعبه مفروض همبسته می‌کند. عمل Read (خواندن)، آخرین مقدار نوشته شده در جعبه‌مشخص را بر می‌گرداند. در صورتی که Read به جعبه‌ای اعمال شود که هرگز نوشته نشده است، مقدار بازگشتی، نامشخص است.

حالت سراسری، مجموعه‌ای از تمام جعبه‌های موجود و مقادیر کنونی اختصاص یافته به آن‌ها است. ویژگی منحصر به فرد جعبه‌ها این است که عملیات آن‌ها حالت سراسری را نیز در بر می‌گیرد: عمل Read به حالت سراسری دسترسی پیدا می‌کند و عمل Create و Write حالت سراسری جدیدی را تولید می‌کنند.

1 - Manipulated
2 - Signature

یادآوری ۱- حالت سراسری فقط به عنوان یک مفهوم مدل‌سازی است. هیچ برنامه‌ی مجزایی که روی ماشین خاصی در حال اجرا است، نمی‌تواند به تمام اجزای حالت سراسری دسترسی داشته باشد. این مشخصه که هر قسمت از سامانه فقط می‌تواند به تعداد کمی از جعبه‌های «محلی»[1] دسترسی داشته باشد، و باید از قسمت‌های دیگر «راه دور»[2] سامانه بخواهد که جعبه‌های راه دور را بخوانند یا بنویسند، مشخصه‌ی سامانه‌های توزیع شده است.

جعبه‌ها که به عنوان نقطه‌هایی دریک دنباله‌ی اجرایی مدل شده‌اند، دلالت بر تصویری از زمان نیز دارند. یک دنباله‌ی اجرایی یک سری از حالت‌های سراسری s_1، s_2 و ... است که در آن هر حالت به جز حالت اول، از حالت قبلی‌اش به وسیله یک تک عمل Create یا یک تک عملWrite مشتق می‌شود.

یادآوری ۲- در رابطه با پردازش موازی، سری حالت‌های سراسری که دنباله‌ی اجرایی برنامه را می‌سازند، در صورت لزوم نمی‌تواند به وسیله متن برنامه معین شوند و ممکن از اجرایی به اجرای دیگر تغییر کنند.

۶-۳ نماد

نماد، ارجاعی به مقداری از نوع داده‌ای خاص (شامل جعبه‌ها و بستارهای رویه) در متن برنامه است. مقادیری که مورد ارجاع قرار می‌گیرند، مقادیری هستند که رویه می‌تواند در طول اجرا به‌طور مستقیم به آن‌ها دسترسی داشته باشد. نمادهای یک رویه خاص در سه دسته‌ی متفاوت قرار دارند:

- نمادهای سراسری که برای ارجاع به مقادیری که به صورت دائمی به رویه همبسته شده‌اند، به کار می‌روند (به عنوان مثال، رویه‌های دیگر، متغیرهای غیر محلی، یا متغیرهای «متعلق به خود[3]»).
- نمادهای محلی که برای ارجاع به مقادیری که فقط در طول تک احضار وجود دارند، به کار می‌روند (به عنوان مثال، متغیرهای «قاب پشته[4]ی» محلی).
- نمادهای پارامتری، پارامترهای صوری هستند که جهت ارجاع به مقادیر پارامترهای واقعی در یک احضار خاص به کار می‌روند.

یادآوری ۱- نمادهای محلی و نمادهای پارامتری یک رویه ممکن است نمادهای سراسری رویه دیگر باشند (برای مثال، رویه‌های تو در تو).

یادآوری ۲- نحوه‌ی ارجاع به مقادیر در متن برنامه‌ای با زبانی خاص، به وسیله قوانین زبان (که شامل قوانین حوزه[5]اش نیز می شود) بیان می‌شود. به عنوان مثال، ابزار ارجاع ممکن است یک شناسه باشد، و همان شناسه می‌تواند مربوط به دو ارجاع متفاوت در زمینه‌های برنامه‌ی متفاوت باشد (به علت قوانین حوزه). لذا از دید این زیربند[6]، شناسه متناظر با دو «نماد» متفاوت خواهد بود.

یادآوری ۳- با انقیاد نمادهای سراسری به جعبه‌ها، این نمادهای سراسری می‌توانند (به صورت غیرمستقیم) به مقادیری که در زمان‌های دلخواه ایجاد شده‌اند، ارجاع کنند، و برای مدت دلخواهی به رویه داده شده همبسته شوند. بنابر این عبارت «همبسته شده به صورت دائمی» محدودیتی برای چیزی که می‌تواند مدل شود، نیست.

۶-۴ تصویر رویه

1 - Local
2 - Remote
3 - Own
4 - Stack frame
5 - Scope
6 - Subclause

تصویر رویه، انتزاعی از متن رویه است. نوع رویه، نمادهای سراسری، محلی، و پارامتری استفاده شده در متن رویه، و الگوریتمی که به وسیله پردازنده اجرا می‌شود در تصویر رویه به صورت ضمنی قرار دارند. چهار عمل بر روی تصویرهای رویه تعریف می‌شوند:

gsyms: Image → Sequence (Symbol)

lsyms: Image → Sequence (Symbol)

psyms: Image → Sequence (Symbol)

spec: Image → Procedure_Type

یادآوری ۱- ترتیب در دنباله‌ای که به وسیله *gsyms* و *lsyms* تولید شده است به ندرت مرتبط هستند، اما ترتیب در دنباله‌ی تولید شده به وسیله *psyms* مهم است (به بند ۶-۱۱ مراجعه شود).

Gsyms نمادهای سراسری تصویر را باز می‌گرداند. *lsyms*، *psyms*، و *spec* به دنباله نمادهای محلی، نمادهای پارامتری و نوع رویه را بر می‌گردانند.

یادآوری ۲- تصویرهای رویه به وسیله پردازنده زبان ایجاد می‌شوند. چگونگی ایجاد یک تصویر رویه خارج از حیطه‌ی این استاندارد ملی است.

۶-۵ همبستگي[1]

همبستگی، هر نگاشتی از یک مجموعه از نمادها به مقادیر است.

A: Symbol → Value

همبستگی‌ها به‌طور معمول ناقص هستند و فقط روی نمادهایی تعریف می‌شوند که در تصویر رویه خاصی استفاده شده‌اند. در صورتی که x یک نماد، y یک مقدار، و A و B همبستگی‌ها باشند:

- $[x \rightarrow y]$ همبستگی را نشان می‌دهد که نماد x را به مقدار y نگاشت می‌کند (و هیچ نماد دیگری را نگاشت نمی‌کند).
- A + B همبستگی را نشان می‌دهد که شرایط زیر را برآورده می‌کند:
  - در صورتی که B بر روی x تعریف شده باشد: $(A+B)(x) = B(x)$
  - در غیر این صورت: $(A+B)(x) = A(x)$
- domain(A) مجموعه نمادهای x ای را نشان می‌دهد که A(x) برای آن‌ها تعریف شده است
- range(A) مجموعه مقادیر {A(x) | x is in domain(A)} را نشان می‌دهد

۶-۶ بستارهاي رويه[2]

بستار رویه یک زوج <I, A> است که در آن I تصویر رویه، و A همبستگی است که نمادهای سراسری I را نگاشت می‌کند (و نه دیگر نمادها را). به طور خاص نمادهای پارامتری و محلی نگاشتی ندارند. بستارهای رویه مقادیر نوع رویه هستند که در انواع داده‌ای مستقل از زبان - ISO/IEC 11404 به آن‌ها اشاره شده است.

1 - Association
2 - Procedure closures

یک بستار رویه *کامل*[1]، بستار رویه‌ی است که در آن تمام نمادهای سراسری تصویر نگاشت شده‌اند.
یک بستار رویه *ناقص*[2]، بستار رویه‌ی است که در آن حداقل یکی از نمادهای سراسری نگاشت نشده باشد.

یادآوری ۱- مثالی از بستار رویه ناقص وقتی است که مقدار رویه A درون رویه B (قبل از این که رویه B فراخوانی شود) قرار داشته باشد. این بستار ناقص است، زیرا در اینجا، مرجع‌های A به متغیرهای محلی B نمی‌توانند قبل از احضار B نگاشته شوند.

یادآوری ۲- بستارهای رویه به‌طور معمول بسته به قوانین زبان برنامه‌نویسی ویژه‌ی به کار رفته، در زمان ترجمه یا در طول اجرا ساخته می‌شوند.

۶-۷ جعبه‌ها، اشاره‌گرها، مقادیر، و انواع داده‌ای

نوع داده‌ای اشاره‌گر، همان‌گونه که در انواع داده‌ای مستقل از زبان - ISO/IEC 11404 تعریف شده است، نوع داده‌ای است که مقادیرش ارجاع‌هایی به دیگر مقادیر است. به صورت خاص‌تر، مقدار یک نوع داده‌ای اشاره‌گر-به-D (D نوع داده است)، ارجاعی به مقداری با نوع داده‌ی D است.
در مدل LIPC، نوع داده‌ی یک جعبه، نوع داده‌ی یک اشاره‌گر است (همان‌گونه که در انواع داده‌ای مستقل از زبان - ISO/IEC 11404 تعریف شده است). هر جعبه مقداری دارد که ارجاعی به مقدار دیگری است. در صورتی که از یک جعبه (به عنوان مثال در نگاشت LIPC)، برای مدل کردن مفهوم «یک متغیر از نوع داده‌ی D» استفاده شده باشد (که برخی از زبان‌ها دارند)، بنابراین مقداری از نوع داده‌ی D را «نگه می‌دارد»[3] (به بند ۶-۲ مراجعه شود) و جعبه، یک مقدار از نوع داده‌ای اشاره‌گر-به-D است.

یادآوری ۱- هستاری که «یک متغیر از نوع داده‌ای D» نامیده می‌شود، نمی‌تواند به‌طور لفظی[4] یک مقدار از نوع داده‌ای D باشد، زیرا چنین مقادیری نمی‌توانند تغییر کنند. هر چیزی بیشتر از یک «آرایه عدد صحیح» می‌تواند به‌طور لفظی یک مقدار از نوع داده‌ای عدد صحیح باشد (زیرا آن‌ها فقط مقادیر منفرد هستند).

در انواع داده‌ای مستقل از زبان - ISO/IEC 11404، «محتوای اشاره‌گر» به عنوان عملگر توصیفی[5] تمام انواع داده‌ای اشاره‌گر تعریف می‌شود. زمانی که این عمل بر روی مقدار P از نوع داده‌ی اشاره‌گر-به-D اعمال شود، نتیجه مقدار V از نوع داده‌ی D است که P به آن ارجاع می‌کند. در مدل LIPC، عملگر متناظری که بر روی یک جعبه اعمال می‌شود، Read است. عملگرهای Create و Write برای جعبه‌ها (به بند ۶-۲ مراجعه شود) عملگرهای توصیفی نیستند، اما به ترتیب متناظر با مواقعی هستند که اشیاء جدیدی از نوع اشاره‌گر ایجاد می‌شود، و زمانی که مقدار V از نوع داده‌ی D ارجاع شده به وسیله یک مقدار P اشاره‌گر-به-D خاص، با مقدار جدیدی جایگزین می‌شود. هر دوی این عملیات در مدل LIPC برای جعبه‌ها نیاز هستند، اگرچه هیچ‌یک در صورت لزوم برای تمام اشیاء از نوع اشاره‌گر در تمام شرایط نیاز نیستند.

---
1 - Complete
2 - Partial
3 - Holds
4 - Literally
5 - Characterizing operator

یادآوری ۲ - عمل Write بر روی جعبه، متناظر با مفهوم «انتساب مقدار[1]» در بسیاری از زبان‌ها است. تغییر مقدار نوع داده‌ی D که یک جعبه به آن ارجاع می‌کند، خود جعبه را تغییر نمی‌دهد بلکه فقط محتویاتش را تغییر می‌دهد. به‌طور دقیق مانند اختصاص یک مقدار جدید به متغیر X که خود X را تغییر نمی‌دهد و X کماکان همان متغیر با همان نام خواهد بود.

یادآوری ۳ - مفهوم «متغیرهای اشاره‌گر[2]» که گاهی به آن «آدرس‌دهی غیرمستقیم» نیز می‌گویند، در برخی از زبان‌ها وجود دارد. انواع داده‌ای مستقل از زبان - ISO/IEC 11404 نوع داده‌ی چنین شیءای را اشاره‌گر-به-اشاره‌گر-به-D[3] تعریف می کند. متغیر اشاره‌گر، به مقداری از نوع داده‌ی اشاره‌گر-به-D ارجاع می‌کند (یعنی شیءای از نوع اشاره‌گر دیگر) که آن اشاره‌گر نیز خود به مقداری از نوع D ارجاع می‌کند. این مورد در مدل LIPC معادل جعبه‌ای است که ارجاعی به جعبه‌ی دیگر (که مقداری از نوع داده‌ای D دارد) را نگه می‌دارد.

به این شکل مدل LIPC از ارسال پارامتر از نوع داده‌ای اشاره‌گر که برخی زبان‌ها به صورت مستقیم یا غیرمستقیم دارند، پشتیبانی می‌کند. همچنین هم استانداردهای LID و هم استانداردهای LIPC، از آدرس‌دهی غیرمستقیم با هر عمقی پشتیبانی می‌کنند.

یادآوری ۴ - در برخی از بحث‌های زبان‌های برنامه نویسی مفهوم «نمونه‌های مقادیر»[4] استفاده می‌شود. در اصطلاحات LIPC، یک نمونه از مقدار می‌تواند به عنوان مقدار نگه داشته شده در یک جعبه تعبیر شود. که اجازه‌ی مدل کردن مواقعی که چندین نمونه از یک مقدار به صورت همگام وجود دارد را می‌دهد. در مدل LIPC بیش از یک هستار می‌توانند به یک جعبه دسترسی داشته باشند (به عنوان مثال، زمانی که به عنوان پارامتر واقعی ارسال شده باشد که پارامتر صوری‌اش از نوع اشاره‌گر است). موجودیت‌هایی که به نمونه‌ی یکسانی از مقدار دسترسی دارند، در صورتی که جعبه تغییر کند (یعنی مقداری که نگه می‌دارد با استفاده از عمل Write تغییر پیدا کند)، هر دو هستار می‌توانند این تغییر را ببینند و بنابر این می‌تواند رفتارهای بعدی هر کدام یا هر دو را تحت تأثیر قرار دهد. در محیط‌هایی که چنین دسترسی همگامی نمی‌تواند به صورت مستقیم پشتیبانی شود، برخی از سازوکارهای تعریف شده در پیاده‌سازی جهت شبیه‌سازی آن باید فراهم شود. به طور مثال به وسیله ایجاد جعبه‌های دو نسخه‌ای، و اطمینان از این‌که هر تغییری در یکی از آن‌ها به صورت خودکار یا در همان لحظه و یا حداقل قبل از این‌که از مقدار جعبه استفاده شود، به دیگری نیز اعمال می‌شود.

نوع داده‌ی رویه، همان‌گونه که در انواع داده‌ای مستقل از زبان - ISO/IEC 11404 تعریف شده است، در حالت کلی یک نوع داده‌ی ترکیبی (اما نه انبوهشی[5]) است که در مشخصاتش شامل انواع داده‌ی تمام پارامترهایش (که تمام پارامترهای مفهومی[6] که برای بازگشت نتایج به رویه‌های «تابع» استفاده می‌شوند را در بر می‌گیرد) می‌شود. در مدل LIPC، یک بستار رویه، هستاری است که نوع داده‌ای آن، نوع داده‌ی رویه LID است، اما مفهوم حالت سراسری را نیز در بر می‌گیرد.

یادآوری ۵- به این وسیله، مطابقت[7] انواع داده‌ای رویه‌ها در مدل LIPC، به صورت خودکار مطابقت تعداد و انواع داده‌ای پارامترها را نیز تضمین می‌کند.

۸-۶ بستار واسط[8]

1 - Value assignment
2 - Pointer variables
3 - Pointer-to-pointer-to-D
4 - Instances of values
5 - Aggregate
6 - Notional
7 - Matching
8 - Interface closure

یک بستار واسط، مجموعه‌ای از اسامی و مجموعه‌ای از بستارهای رویه با نگاشت بین آن‌ها است. و به وسیله یک همبستگی که مجموعه‌ای از اسامی را به بستارهای رویه نگاشت می‌کند، مدل می‌شود.

یادآوری - برای مثال، اگر Sue، Mary و Sam، اسامی رویه باشند (نمادها)، و X و Y و Z بستارهای رویه باشند، عبارت:

$$I = [\text{Sue} \rightarrow X] + [\text{Mary} \rightarrow y] + [\text{Sam} \rightarrow Z]$$

یک بستار واسط است. و domain(I) = {Sue, Mary, Sam} (به بند ۶-۵ مراجعه شود). بنابر این domain(I)، مجوعه‌ی اسامی رویه در بستار واسط I است. بستار رویه‌ی که در I، Mary نام دارد، با I(Mary) نشان داده می‌شود.

۶-۹ نوع واسط

نوع واسط مجموعه‌ای از اسامی و مجموعه‌ای از انواع رویه با نگاشت بین آن‌ها است. و به وسیله یک همبستگی که مجموعه‌ای از انواع رویه را به اسامی نگاشت می‌کند، مدل می‌شود.

یادآوری ۱ - یک نوع واسط، یک نوع داده‌ای نیست.

یادآوری ۲ - به طور مثال، اگر Sue، Mary و Sam، اسامی رویه باشند، و X و Y و Z به ترتیب بستارهای رویه برای انواع رویه متناظر XT، YT، و ZT باشند، آنگاه عبارت: $IT = [\text{Sue} \rightarrow XT] + [\text{Mary} \rightarrow YT] + [\text{Sam} \rightarrow ZT]$

نوع واسط متناظر بستار واسط I است که در زیربند قبل آورده شد.

۶-۱۰ مشخصات[1]

مدل LIPC عمل *spec* را بر روی تصاویر رویه جهت بازگرداندن نوع رویه تصویر تعریف می‌کند. بنابر این،

spec: Image → Procedure_Type

spec برای بستارهای رویه به وسیله

spec: Procedure_Closure → Procedure_Type

spec (<image,assoc>) = spec (image)

تعمیم داده می‌شود و spec برای بستارهای واسط به وسیله عبارات زیر تعمیم می‌یابد که در آن، P، procedure_type و I، interface_closure است.

spec: Interface_Closure → Interface_Type

For $I = [name_1 \rightarrow P_1] + \ldots + [name_n \rightarrow P_n]$,

$spec(I) = [name_1 \rightarrow spec(P_1)] + \ldots + [name_n \rightarrow spec(P_n)]$

عمل spec بر روی interface_closure، یک همبستگی بین اسامی و انواع رویه را بر می‌گرداند.

۶-۱۱ احضار رویه پایه[2]

احضار رویه پایه، عملی است بر روی بستارهای رویه کامل، که با عبارت زیر توصیف می‌شود:

invoke: Procedure_Closure * Sequence(Value) → Status * Sequence(Value)

1 - Specification
2 - Basic procedure invocation

که در آن Status مجموعه‌ای از شناسه‌های پایان‌دهی است (به بند ۳-۹ از ISO/IEC 11404:1996 مراجعه شود). اولین دنباله از مقادیر، پارامترهای ورودی احضار را نشان می‌دهد. دومین دنباله از مقادیر، مقادیری را نشان می‌دهد که نتیجه‌ی احضار هستند. status شرط پایان‌دهی را نشان می‌دهد، که شامل پایان‌دهی عادی نیز می‌شود.

به کارگیری احضار برای بستار رویه <تصویر ، همبستگی> و مقادیر ورودی $<V_1,\ldots V_n>$ منجر به عملیات زیر می‌شود:

Let $<A_1, \ldots A_n>$ = psyms (Image)

$<L_1, \ldots L_m>$ = lsyms (Image)

For i = 1 to m, do

LB_i = create ()

Define invocation association Q by:

$Q = \text{Association} + [A_1 \rightarrow V_1] + \ldots + [A_n \rightarrow V_n]$

$+ [L_1 \rightarrow LB_1] + \ldots + [L_m \rightarrow LB_m]$

Then

"Execute Image in the context of association Q"

اجرای یک تصویر در زمینه، یک مفهوم اولیه است که به وسیله پردازنده (یا استاندارد) زبان برنامه‌نویسی برای زبانی که تصویر در آن نوشته شده است، تعریف می‌شود. زمانی که اجرا پایان پیدا می‌کند، مقداری در Status * Sequence(Value) نتیجه خواهد شد و همبستگی Q از بین می‌رود.

یادآوری – جعبه‌هایی که در شکل‌گیری Q ایجاد شده‌اند دیگر قابل دسترسی نیستند مگر این‌که زبان برنامه‌نویسی به آن‌ها اجازه بدهد به نحوی «باز گردانده» شوند (به بند ۶-۱۳ مراجعه شود).

۶-۱۲ درستي نوع[1]

به کار بردن عمل احضار بر روی هر بستار رویه‌ی مانند C و هر دنباله‌ای از مقادیر ورودی $<V_{1,\ldots}V_n>$ معنا ندارد. احضار فقط در صورتی که نوع پارامترهایش صحیح باشند معنا دارد. فرض می‌کنیم که spec(c) به صورت زیر باشد:

PROCEDURE (a_1: AT_1, … a_{an}: AT_{an})

RETURNS (r_1: RT_1, …r_{rn}: RT_{rn})

SIGNALS (E_l, ... E_{en})

که در آن a_1 تا a_{an} پارامترهای ورودی (به ترتیب)، r_1 تا r_{rn} پارامترهای خروجی (به ترتیب)، و E_1 تا E_{en} پایان‌دهی‌های غیرعادی هستند.

احضار C بر روی $<V_1, \ldots V_n>$ در صورتی از نوع صحیح است که n=an باشد (تعداد پارامترها درست باشد)

و

1 - Type correctness

For i = 1 to n,

V_i is a value of type AT_i (انواع پارامترها صحیح باشند)

زمانی که احضار C بر روی $< V_1, \ldots V_n >$ پایان پیدا می‌کند، نتیجه‌ای به شکل:

$< status, < W_1, \ldots W_m > >$

تولید می‌کند. در صورتی که بستار C عضوی از نوع رویه‌ی باشد که به آن نگاشته شده است، اطلاعات زیر برای نتایج بالا به دست می‌آید.

status = "normal" or status = E_i for some i in 1 .. en

If status = "normal" then

m = rn, and

W_j is a value of type RT_j (for all j in 1 .. m)

ElseIf status = E_i, and E_i is declared to have structure

f_1: FT_1, ... f_{fn}: FT_{fn}

then

m = fn, and

W_j is a value of type FT_j (for all j in 1 .. m)

۶-۱۳ همبسته‌ها[1]

برای این‌که بتوان مجموعه‌ی متغیرهای سراسری که در دو رویه مشترک است را بررسی کرد، یا مستعارسازی[2] اشاره‌گر (یا پارامتر) را تعریف کرد، لازم است بدانیم که در چه زمانی یک مقدار، «ارجاع شده به وسیله»[3] یا «قابل دستیابی از»[4] مقدار دیگر است. X یک *همبسته ساده* از Y است در صورتی که X بتواند از Y با دنبال کردن اشاره‌گرها یا استخراج کردن عناصرِ مقادیر انبوهشی[5] به دست آید. X یک *وابسته تعمیم یافته*[6] از Y است در صورتی که X بتواند از Y با ترکیب اعمال بالا و رویه‌های احضار شده به دست آید.
دو بند بعدی این دو مفهوم را به طور صوری بیان می‌کنند.

۶-۱۳-۱ همبسته‌های ساده

مفهوم همبسته‌های ساده در دو تابع نهفته است.
اولین تابع، همبسته بلافصل[7] است. IAssoc(x) به شکل زیر تعریف می‌شود:

IAssoc: Value → Set(Value)

1 - Associates
2 - Aliasing
3 - Referenced by
4 - Accessible from
5 - Aggregate
6 - Generalized associate
7 - Immediate Associates

- در صورتی که x مقدار یک نوع غیر-انبوهشی[1] تعریف شده در انواع داده‌ای مستقل از زبان – ISO/IEC 11404 باشد، IAssoc(x) یک مجموعه‌ی تهی خواهد بود.
- در صورتی که x مقدار یک نوع انبوهشی تعریف شده در انواع داده‌ای مستقل از زبان – ISO/IEC 11404 باشد، IAssoc(x) مجموعه‌ی تمام عناصر انبوهشی خواهد بود.
- در صورتی که x جعبه‌ای باشد که در آن لحظه مقدار v را نگه می‌دارد،

IAssoc(x) = مجموعه‌ای شامل تک مقدار v است.

- در صورتی که x بستار رویه باشد،

IAssoc(x) = یک مجموعه‌ی تهی است.

دومین تابع همبسته، همبسته متعدی[2] است:

Assoc: Value → Value

برای هر مقدار x، Assoc(x) کوچکترین مجموعه‌ای است که شرط زیر برای آن برقرار باشد:

X در Assoc(x) قرار دارد

اگر y در Assoc(x) قرار داشته باشد، آنگاه تمام عناصر IAssoc(y) در Assoc(x) هستند.

یادآوری ۱- پر واضح است که Assoc(x) شامل تمام مقادیری می‌شود که می‌توانند از x با به کارگیری عملیات استخراج مختلف بر روی انبوهشی‌ها و عملیات خواندن بر روی جعبه‌ها استخراج شوند. از آن‌جایی که خواندن به حالت کنونی وابسته است، Iassoc و Assoc نیز به حالت کنونی وابسته می‌شوند.

زمانی که یک بستار رویه <I,A> بر روی ورودی‌های < V1, ... Vn> احضار می‌شود، به تمام مقادیر در

$range(A) \cup \{V_1,\dots V_n\}$

دسترسی مستقیم و بلافصل دارد. و (با برخی از محاسبات) به تمام مقادیر در

$Z = Assoc\ (range(A) \cup \{V_1,\dots V_n\}\ )$

دسترسی مستقیم داد.

احضار <I,A> بر روی <V1,…Vn> به صورت بالقوه می‌تواند هر جعبه‌ای در Z (و نه مابقی را) را بخواند یا بنویسد.

یادآوری ۲ - این کار می‌تواند جعبه‌های جدید را نیز ایجاد کند.

یادآوری ۳ - مجموعه‌ی Z، شامل مقادیری که فقط به وسیله احضار بستارهای رویه دیگر می‌توان به آن‌ها دسترسی داشت، نمی‌شود.

1 - Non-aggregate
2 - Transitive Associates

۶-۱۳-۲ همبسته‌های تعمیم‌یافته[1]

مفهوم همبسته‌های تعمیم‌یافته شامل مقادیری می‌شود که می‌توانند با کمک رویه‌های دیگر به دست آیند. در این‌جا نیز دو تابع نیاز است.

همبسته‌های بلافصل تعمیم‌یافته[2] به شکل زیر تعریف می‌شود:

$$GIAssoc: Value \rightarrow Value$$

در صورتی که x بستار رویه $<I, A>$ باشد و

$$GIAssoc(x) = range(A)$$

در صورتی که x هر مقدار دیگری باشد.

$$GIAssoc(x) = IAssoc(x)$$

همبسته‌های متعدی تعمیم‌یافته[3] نیز به شکل زیر تعریف می‌شود:

$$GAssoc: Value \rightarrow Value$$

برای هر مقدار x، GAssoc(x) کوچکترین مجموعه‌ای است که موارد زیر را برآورده می‌کند:

X در GAssoc(x) قرار دارد

اگر y در GAssoc(x) باشد، آنگاه تمام عناصر GAssoc(y) در GAssoc(x) هستند.

یادآوری - پر واضح است که GAssoc(x) تمام مقادیری که می‌تواند از x با به کارگیری اعمال استخراج مختلف بر روی انبوهشی‌ها، خواندن بر روی جعبه‌ها، و احضار رویه استخراج شوند را در بر می‌گیرد.

زمانی که بستار رویه $<I,A>$ بر روی ورودی‌های $<V_1,\ldots V_n>$ احضار می‌شود، و:

$$GZ = GAssoc\ (range(A)\ U\ \{V_1, \ldots\ V_n\}\ )$$

به کمک بستارهای رویه دیگر، این احضار می‌تواند به هر مقداری در GZ دسترسی داشته باشد. احضار $<I, A>$ بر روی $<V_1,\ldots V_n>$ تنها می‌تواند به عناصری از حالت سراسری دسترسی پیدا کند یا آنها را تغییر دهد که جعبه‌هایی در GZ باشند.

فرض بر این است که یک بستار رویه $<I,A>$ در صورتی می‌تواند بستار رویه دیگر $<J,B>$ را احضار کند که $<J,B>$ یکی از موارد زیر باشد:

۱- در range(A) (رایج‌ترین شرط) باشد

۲- از یک پارامتر ورودی قابل دستیابی باشد،

۳- J در محدوده‌ی احضار همبستگی $<I,A>$ باشد و B بتواند از مقادیر قابل دستیابی ساخته شود.

۶-۱۴ زمینه‌هاي احضار و اجرا[4]

1 - Generalized Associates
2 - Generalized Immediate Associates
3 - Generalized Transitive Associates
4 - Execution and invocation contexts

زمینه‌ی اجرای بستار رویه <تصویر و همبستگی>، مجموعه‌ی تمام جعبه‌ها در ssoc(range(Association)) است. زمینه‌ی احضارِ یک احضار خاص از بستار رویه <تصویر و همبستگی>، مجموعه‌ای از تمام جعبه‌های Assoc(range(Q)) است که در آن Q، همبستگی احضارِ مربوط به این احضار از <تصویر و همبستگی> است.

یادآوری – هم زمینه‌ی اجرا و هم زمینه‌ی احضار در طول اجرای تصویر به مرور زمان می‌توانند تغییر کنند.

۱۵-۶ ترجمه‌ي پارامتر

زمانی که نیاز است احضار رویه‌ی بین زمینه‌های اجرا رفت و برگشت کند، ممکن است نتوان به‌طور مستقیم پارامتر را بین این زمینه‌ها ارسال کرد و مقادیر را بازگرداند. دو مثال زیر را در نظر بگیرید:

در مثال اول، برنامه‌ی نوشته شده در زبان برنامه‌نویسی L1 رویه‌ی در زبان L2 را فراخوانی می‌کند. در صورتی که L1 و L2 انواع داده‌ای متفاوتی داشته باشند، این فراخوانی ممکن است نیاز داشته باشد که مقادیر ورودی L1 را به معادل‌هایشان در L2 ترجمه کند. همچنین در زمان بازگشت نیز ممکن است یک ترجمه‌ی معکوس نیاز باشد.

در مثال دوم، یک برنامه ممکن است رویه نوشته شده به زبان یکسان (بنا بر این هیچ ترجمه‌ی نوع داده‌ای نیاز نیست)، اما در فضای آدرس جداگانه‌ای را فراخوانی کند. فرض کنید اشاره‌گرها به نحوی پیاده‌سازی شده‌اند که آن‌ها را به فضای آدرس مشخصی وابسته می‌کند (که مورد رایجی است). لذا هر اشاره‌گری در مقادیر ورودی به فضای آدرس رویه کارخواه وابسته خواهد بود. این اشاره‌گرها باید به شکل یکنواختی با اشاره‌گرهای «معادلی»[1] که به فضای آدرس رویه وابسته هستند، جایگزین شوند. در این‌جا نیز امکان دارد در زمان بازگشت هم یک ترجمه‌ی معکوس نیاز باشد.

ترجمه‌ی پارامتر ممکن است باعث از دست دادن اطلاعات شود (برای مثال هنگام ترجمه بین اشکال مختلف ممیز شناور)، و می‌تواند روابط اشتراک را بر هم بریزد (برای مثال، هنگام حرکت دادن اشاره‌گرها بین فضاهای آدرس). از آنجایی که برنامه‌نویسان می‌توانند این تأثیرات را ببینند، مدل LIPC روشی را برای مدیریت آن‌ها تعریف می‌کند.

ترجمه‌ی پارامتر به روشی که در ادامه می‌آید، مدل می‌شود. اگر C یک بستار رویه دلخواه باشد، و TF و TB بستارهای رویه‌ی باشند که ترجمه‌های پارامتر را برای C انجام می‌دهند، آنگاه

wrap (TF, C, TB)

را بستار رویه‌ی تعریف می‌کنیم که (زمانی که احضار می‌شود) موارد زیر را انجام می‌دهد:

۱- TF را جهت ترجمه‌ی ورودی‌ها احضار می‌کند،

۲- C را با پارامترهای ترجمه شده، احضار می‌کند، و

۳- TB را جهت ترجمه‌ی دوباره‌ی مقادیر برگشتی احضار می‌کند.

در ادامه توضیح می‌دهیم که چگونه تابع wrap در مدل کردن فراخوانی‌های همراه با گذر بین زمینه‌های اجرا، کمک می‌کند. X1 و X2 را زمینه‌های اجرا در نظر می‌گیریم.

1 - Equivalent

یادآوری ۱- مهم نیست که زمینه‌ی اجرا چیست، فقط مهم این است که ترجمه‌ای برای فراخوانی از یکی به دیگری لازم است.

فرض می‌کنیم C1 بستار رویه‌ی باشد که رویه مقصد را در زمینه‌ی ذاتی خودش (X1) نشان می‌دهد. و
C2 = wrap (TF, Cl, TB)

بستار رویه‌ی است که به‌طور واقعی در زمینه‌ی X2 فراخوانی می‌شود. در بسیاری از موارد، فراخوانی C2 اثرات متفاوت قابل توجهی نسبت به فراخوانی C1 خواهد داشت.

تعریف دقیق‌تری از wrap به شکل زیر است:

wrap: Procedure_Closure * Procedure_Closure * Procedure_Closure

→ Procedure_Closure

wrap (TF, C, TB) = <IM, [pre→TF] + [main→C] + [post→TB]>

جهت راحتی کار فرض کنید که TF و TB یک ورودی Sequence(Value) تک می‌گیرند و یک خروجی Sequence(Value) تک تولید می‌کنند. این امر به TB به طور خاص اجازه می‌دهد که روی دنباله‌های خروجی با طول‌های متفاوت احضار شود.

هنگامی که بستار رویه wrap(TF,C,TB) روی دنباله‌ی ورودی V احضار شود، تصویر IM باعث می‌شود گام‌های زیر رخ دهند:

1. TF is invoked on <V>, producing <E, W>

   (1.1) If E ¬= "normal", IM terminates with <E, W>

   (1.2) If E = "normal", W is a singleton $<W_1>$

2. C is invoked on W_1, producing <F, X>

3. TB is invoked on <X>, producing <G, Y>

   (3.1) If G ¬= "normal" , IM terminates with <G, Y>

   (3.2) If G = "normal" , Y is a singleton $<Y_1>$

4. IM terminates with $<F, Y_1>$

استفاده از بستارهای رویه برای انجام ترجمه‌های پارامتر و نتیجه، اجازه می‌دهد مدل با تمام قدرت محاسباتی جهت بیان کردن این ترجمه‌ها استفاده شود. TF و TB می‌توانند با یکدیگر از طریق جعبه‌های مشترک یکدیگر ارتباط برقرار کنند، و می‌توانند به قسمت‌های دلخواه دیگر حالت سراسری، در صورتی که نگاشت‌های همبستگی آنها به‌طور متناظر تعریف شده باشند، دسترسی داشته باشند. انتظار می‌رود که TF و TB به‌طور کامل ساده باشند.

یادآوری ۲- TF و TB تنها مکان‌هایی هستند که Value (اجتماع تمام انواع داده‌ای) در یک زمینه‌ی مفهومی استفاده می‌شود.

مثال: مدل یک فراخوانی رویه راه دور از فضای آدرس کارخواه[1] (CAS) به فضای آدرس کارساز[2] (SAS). فرض می‌کنیم P رویه‌ی در SAS، MC کد مرتب کردن سمت کارخواه، و UC کد نامرتب کردن سمت کارخواه است. MS و US کدهای متناظر سمت کارساز هستند. بستار رویه:

PW = wrap (US, P, MS)

رویه P را آن‌گونه که به دنیای بیرون ارسال می‌شود، نشان می‌دهد. PW داده‌ی «wire format» را به عنوان ورودی می‌گیرد و داده «wire format» را به عنوان خروجی بر می‌گرداند.

بستار رویه:

PC = wrap (MC, PW, UC)

رویه P را آن‌گونه که به CAS وارد شده است، نشان می‌دهد. ورودی‌ها و خروجی‌های PC برای CAS مناسب هستند.

۱۶-۶ تعریف رویه‌های ترجمه

رویه‌های ترجمه به‌طور معمول نیاز دارند که یک مقدار ترکیبی V را بگیرند و فقط بخش مشخصی از آن را جایگزین کنند (مابقی V را به شکل اصلی خود باقی می‌گذارند). به طور مثال جایگزین کردن تمام جعبه‌ها در V با جعبه‌های جدید، با حفظ ساختار اشتراک V. بیان این مورد به وسیله یک الگوریتم تا حدی پیچیده است. اما برخی از مفاهیم وجود دارند که می‌توانند به توصیف نتیجه‌ی مورد نظر کمک کند (جزئیات الگوریتم به پیاده‌سازها واگذار می‌شود).

تعاریف زیر در این استاندارد ملی استفاده نمی‌شوند، اما به کوتاه کردن تعاریف در استانداردهای انقیاد، کمک خواهند کرد.

در صورتی که T نگاشتی از مقادیر به مقادیر باشد:

$$T: Value \rightarrow Value$$

در صورتی که برای تمام مقادیر v از نوع داده‌ی Q، $T(v) = v$ باشد، T یک *شناسه* بر روری نوع داده‌ی Q خواهد بود. فرض می‌کنیم که F یک عمل توصیفی از نوع داده‌ی Q، و F' یک عمل توصیفی از نوع داده‌ی Q' که تعداد پارامترهای آن با F یکسان است. T، F را *به* F' *نگاشت می‌کند* اگر برای تمام مقادیر $v_1,\ldots v_n$ در دامنه‌ی ورودی F، داشته باشیم:

$$T (F(v_1,\ldots v_n)) = F' (T(v_1),\ldots T(v_n))$$

اگر T تمام عملیات توصیفی Q را به متناظرهایشان در Q' نگاشت کند، می‌گوییم «T، Q را *به* Q' *نگاشت می‌کند*».

در صورتی که T هر عمل توصیفی Q را به خودش نگاشت کند، نوع داده‌ی Q را حفظ می‌کند[3].

1 - Client Address Space
2 - Server Address Space
3 - Preserves

یادآوری - به عنوان مثالی از نحوه‌ی استفاده از مفاهیم بالا، فرض کنید که می‌خواهیم رویه ترجمه TF را برای جایگزین کردن تمام جعبه‌های موجود در مقدار V با جعبه‌های جدید (در حالی که ساختار اشتراک در V حفظ می‌شود)، تعریف کنیم. TF که بر روی دنباله‌ی ورودی V احضار می‌شود به شکل زیر عمل می‌کند:

1. Compute the set

   $\{B_1, ... Bn\} = Assoc (V) \cap Boxes$

2. For i = 1 to n,

   C_i = create ()

3. Let Z be a function Z: Value → Value satisfying

   For any box B_i, $Z(B_i) = C_i$

   Z preserves all aggregate datatypes except Box

   Z is an identity on all non-aggregate datatypes

4. Finally, set TF(V) = Z(V).

۷ نشانه‌گذاري تعريف واسط

نشانه‌گذاری تعریف واسط ابزاری است که در این استاندارد ملی جهت مشخص کردن اعلان‌های رویه‌ها، پارامترهای رویه، انواع داده‌ای، و صفات تعریف شده است. این نشانه‌گذاری از انواع داده‌ای که در انواع داده‌ای مستقل از زبان – ISO/IEC 11404 تعریف شده‌اند، پشتیبانی می‌کند.

یادآوری – برای این‌که پردازنده زبانی با این بند از این استاندارد ملی انطباق داشته باشد (به بند ۴ مراجعه شود)، باید انقیادی بین سازوکار فراخوانی رویه خودش و IDN تعریف شده در این بند را مشخص کند. این انقیاد بسته به حالت انطباقی (کارخواه، کارساز، یا هر دو) که نیاز است، باید نگاشت‌های ورودی و/یا خروجی انواع داده‌ای آن زبان به انواع داده‌ای تعریف شده در انواع داده‌ای مستقل از زبان – ISO/IEC 11404 را در برگیرد.

۷-۱ قراردادهاي تعريفي

۷-۱-۱ مجموعه‌ی نویسه‌ها[1]

حرف (letter)	a b c d e f g h I j k l m n o p q r s t u v w x y z
رقم (digit)	0 1 2 3 4 5 6 7 8 9
خاص (special)	() {} <>
	(parentheses) (braces) (angle-brackets) (full stop) (comma)
	- * / = ; :
	(colon) (semicolon) (equals) (solidus) (asterisks) (minus)
خط تیره (hyphen)	-
آپوستروف (apostrophe)	'
نقل قول (quote)	"
گریز (escape)	!
فاصله (space)	

1 - Character Set

نویسه‌ی فاصله باید مقید به عضو «space» از استاندارد ISO/lEC 10646-1:1993 شود، اما فقط درون الفاظ نویسه‌ای[1] و الفاظ رشته‌ای[2] معنا دارد.
نویسه‌ی مقید[3]، یک حرف، رقم، خط تیره، خاص، آپوستروف، فاصله، یا نقل قول است. یک نویسه‌ی مقید (به استثناء حروف) باید با عضوی که نماد متناظری در مجموعه‌ی نویسه‌های مشتق‌شده از استاندارد ISO/IEC 10646-1:1993 دارد، همبسته شود.
در هر پیاده‌سازی، نویسه‌ی مقید و نویسه‌ی گریز باید به اعضای خاصی از مجموعه‌ی نویسه‌های پیاده‌سازی همبسته شوند.
نویسه‌ی افزوده[4]، نویسه‌ای است که در این استاندارد ملی تعریف نشده است. یک نویسه‌ی افزوده، هر عضو دیگری از مجموعه‌ی نویسه‌های پیاده‌سازی است که به عضوی که نماد متناظری در مجموعه‌ی نویسه‌های استاندارد ISO/IEC 10646-1:1993 دارد، مقید شده است.

۲-۱-۷ نحوِ صوری

این استاندارد ملی یک نمایش صوری جهت اعلان و تشخیص نوع داده تعریف می‌کند. نشانه‌گذاری زیر (که از شکل Backus-Naur گرفته شده است) در تعریف این نمایش صوری استفاده شده است. در این بند، کلمه‌ی *نشانه‌ها*[5]، نویسه‌هایی را نشان می‌دهد که در تعریف سازوکار صوری به کار رفته‌اند. و کلمه‌ی *نویسه*، به نویسه‌هایی اشاره دارد که در شکل‌دهی رویه و اعلان‌ها و شناسایی‌های نوع داده استفاده می‌شوند.
یک نماد پایانی[6] دنباله‌ای از نویسه‌هایی است که با وقوع دو نشان-نقل قول (") محدوده‌شان مشخص می‌شود. اولین نشان-نقل قول قبل از اولین نویسه در نماد پایانی و دومین نشان پس از آخرین نویسه در نماد پایانی می‌آید. یک نماد پایانی وقوع یک دنباله از نویسه‌ها را نشان می‌دهد.
یک نماد غیرپایانی[7] دنباله‌ای از نشانه‌هایی است که یا یک حرف یا یک نشانه‌ی خط تیره (-) هستند و با اولین نشانه‌ای که حرف یا خط تیره نباشد خاتمه پیدا می‌کند. یک نماد غیرپایانی هر دنباله‌ای از نمادهای پایانی است که تولید آن نماد غیرپایانی را امکان‌پذیر می‌سازند. برای هر نماد غیرپایانی به‌طور دقیق یک ساخت[8] IDN وجود دارد. نمادهای غیرپایانی در متن این استاندراد ملی با حروف مورب مشخص می‌شوند.
یک دنباله‌ی تکراری[9]، دنباله‌ای از نمادهای پایانی و/یا غیرپایانی است که در بین یک نشانه‌ی آکولاد باز ({) ویک نشانه‌ی آکولاد بسته (}) قرار دارند. دنباله‌ی نمادهایی که این چنین محاط شده‌اند می‌توانند به هر تعدادی در مکانی که دنباله‌ی تکرار شدنی قرار دارد، رخ دهند (اما نیازی نیست به‌طور حتم رخ دهند).

1 - Character-literals
2 - String-literals
3 - Bound-character
4 - Added-character
5 - Marks
6 - Terminal symbol
7 - Non-terminal symbol
8 - Production
9 - Repeated sequence

یک دنباله‌ی اختیاری[1]، دنباله‌ای از نمادهای پایانی و/یا غیرپایانی است که بین کروشه‌ی باز ([) و کروشه‌ی بسته (]) قرار دارند. دنباله‌ی نمادهایی که این چنین محاط شده‌اند می‌توانند یکبار در مکانی که دنباله‌ی اختیاری قرار دارد، رخ دهند (اما نیازی نیست به‌طور حتم رخ دهند).

یک دنباله‌ی جایگزین[2]، دنباله‌ای از نمادهای پایانی و/یا غیرپایانی است که قبل از آن یک نشانه‌ی سرکش عمودی (|) می‌آید و با یک نشانه‌ی سرکش عمودی یا نشانه‌ی توقف-کامل (.) خاتمه پیدا می‌کند. توالی نمادهایی که این چنین محاط شده‌اند می‌توانند به جای دنباله‌ی نمادهای قبل از اولین سرکش عمودی بیایند.

یک ساخت، دنباله‌های معتبرِ نمادهایی که یک نماد غیرپایانی نشان می‌دهد را تعریف می‌کند. یک ساخت ساده به شکل زیر است:

non-terminal-symbol = valid-sequence.

که در آن valid-sequence، هر دنباله‌ای از نمادهای پایانی، نمادهای غیرپایانی، دنباله‌های اختیاری، دنباله‌های تکراری و دنباله‌های جایگزین است. نشانه‌ی علامت-مساوی (=)، نماد غیرپایانی را که تعریف می‌شود از دنباله‌ی معتبری که تعریفش را نشان می‌دهد، جدا می‌کند. نشانه‌ی توقف کامل، دنباله‌ی معتبر را پایان می‌دهد.

۷-۱-۳ فضای خالی[3]

دنباله‌ای از یک یا تعداد بیشتری از نویسه‌های فاصله، به جز درون لفظ نویسه‌ای[4] یا لفظ رشته‌ای[5]، به عنوان فضای خالی در نظر گرفته می‌شود. هر کاربردی از این استاندارد ملی ممکن است نویسه‌ها یا دنباله‌ی نویسه‌های دیگری (به‌عنوان مثال جدول‌بندهای[6] افقی یا عمودی، پایان خط و شاخص‌های صفحه و غیره) را به عنوان فضای خالی تعریف کند.

یک توضیح[7]، هر توالی از نویسه‌هایی است که با دنباله‌ی «*/» آغاز می‌شوند و با اولین وقوع دنباله‌ی «/*» پس از آن، پایان می‌یابند. هریک از نویسه‌های توضیح به عنوان فضای خالی در نظر گرفته خواهند شد.

هر دو شیءای که پشت سر هم می‌آیند، ممکن است به وسیله فضای خالی، بدون تأثیر گذاشتن بر تفسیر ساختار نحوی، از هم جدا شوند. فضای خالی نباید درون اشیاء لغوی[8] ظاهر شود.

۷-۲ اعلان‌هاي نوع واسط

interface-type = "interface" [interface-synonym ":"]

[interface-identifier] "begin" interface-body "end".

interface-synonym = identifier.

interface-identifier = object-identifier.

---

1 - Optional sequence
2 - Alternative sequence
3 - Whitespace
4 - Character-literal
5 - String-literal
6 - Tabulators
7 - Comment
8 - Lexical

interface-body = {import} {declaration ";"}.

declaration = value-decl | type-decl | procedure-decl | termination-decl.

یادآوری – یک تعریف نوع واسط شامل اعلان هستارهای واسط مختلف مانند ثابت‌ها، انواع داده‌ای، مؤلفه‌های انواع تولید شده (برای مثال فیلدهای یک رکورد) و غیره می‌شود. این اعلان‌ها یک شناسه را به هستار واسط داده شده در اعلان ربط می‌دهند. کاربرد این شناسه، پیشامد تعریفی[1] آن نامیده می‌شود. زمانی که این شناسه در جای دیگری در تعریف نوع واسط استفاده شود، به هستار مربوط به پیشامد تعریفی آن ارجاع می‌کند. جهت اجتناب از ابهام در این‌که یک شناسه‌ی ارجاع به کدام هستار ارجاع می‌کند، قوانین حاکم بر یکتایی تعریف شناسه‌ها و قوانین حاکم بر چگونگی تعیین شناسه‌های ارجاع در بندهای مناسب آورده شده‌اند.

interface-synonym در اعلان *interface-type* یک اسم اختیاری قابل خواندن به وسیله انسان برای یک نوع واسط است. *interface-identifier* این ساخت، یک *object-identifier* است که به صورت یکتا تعریف نوع واسط را مشخص می‌کند.

interface-synonym ها، باید در *interface-type* بلافصل‌شان، یکتا باشند.

۷-۲-۱ مرجع‌های نوع

در بدنه‌ی یک واسط، یک *identifier* که در تعریف نوع واسط برای ارجاع به یک *type-specifier* استفاده می‌شود، *type-reference* نامیده می‌شود (به بند ۷-۷ مراجعه شود).

یک *type-reference* با یک *type-decl*، در صورتی مطابقت می‌کند که *type-identifier* از *type-decl* با مؤلفه‌ی *identifier* از *type-reference* یکسان باشد. قوانین زیر بر استفاده‌ی *type-reference* در یک *interface-type* حاکم است.

در صورتی که مؤلفه‌ی *interface-synonym* یک *type-reference* وجود نداشته باشد، آنگاه *type-reference* باید با یک *type-decl* در *interface-type* بلافاصله در بر گیرنده یا با یک *type-decl* که به *interface-type* بلافاصله در بر گیرنده (یا به صورت صریح با یک *import-symbol* یا به صورت ضمنی با وارد کردن کل تعریف نوع واسط) وارد شده است، مطابقت کند. در صورتی که *type-reference* با *type-decl* در *interface-type* که بلافاصله آن را در بر می‌گیرد، مطابقت کند، به *type-specifier* بلافصلِ آن *type-decl* ارجاع می‌کند. در غیر این صورت، *type-reference* باید حداکثر با یک *type-decl* وارد شده مطابقت کند، و *type-reference* به *type-specifier* بلافصل آن *type-decl* ارجاع خواهد کرد.

یادآوری – اگر *object-identifier* یک *type-decl* وارد شده با *object-identifier* تعریف شده‌ی *interface-type* بلافصل یا *object-identifier* یک *type-decl* وارد شده از یک تعریف نوع واسط متفاوت، یکسان باشد، فقط با استفاده از *interface-synonym* مربوطه‌ی آن می‌تواند مورد ارجاع قرار گیرد.

در صورتی که مؤلفه‌ی *interface-synonym* از *type-reference* وجود داشته باشد، *type-reference* باید با یک *type-decl* در تعریف نوع واسطی که به وسیله *interface-synonym* مشخص شده است، مطابقت کند. *type-reference* به *type-specifier* بلافصل این *type-decl* ارجاع می‌کند.

1 - Defining occurrence

۷-۲-۲ مرجع‌های مقدار

در بدنه‌ی یک واسط، یک *identifier* که در تعریف نوع واسط که جهت ارجاع به یک مقدار استفاده شده است، *value-reference* نامیده می‌شود. یک *value-reference* باید به یکی از موارد زیر ارجاع کند:

۱- یک *value-expression* که در یک *value-decl* استفاده شده است، یا

۲- یک *enumeration-identifier*، یا

۳- یک *field* در *record-type*، یا

۴- *parameter* صوری یک *procedure-decl*، *procedure-type*، یا *termination-decl*، یا

۵- یک *return-arg* در *procedure-decl* یا *procedure-type* ، یا

۶- *formal-value-parm* یک *parameterized-type-decl*،

مقدار یک *value-reference*، ممکن است در صورتی که به یک *value-expression* یا *enumeration-identifier* ارجاع کند، به صورت ثابت شناخته شده باشد. در غیر این صورت در زمان احضار یا پایان‌دهی رویه تعیین می‌شود.

۷-۳ اعلان‌هاي وارد كردن

```
import = "imports" ["("import-symbol-list")"] "from"
    [interface-synonym ":"] object-identifier.
import-symbol-list = import-symbol {"," import-symbol}.
import-symbol = identifier.
```

برای این‌که *interface-body* جاری بتواند به *identifier* های تعریف شده در اعلان‌های نوع واسط دیگر ارجاع کند، باید از اعلان *import* استفاده کرد. *object-identifier* در عبارت *import*، *interface-identifier* مربوط به تعریف نوع واسطی است که در آن نمادها تعریف شده‌اند. در صورتی که *interface-synonym* در ساخت *import* وجود داشته باشد، ممکن است در محدوده‌ی *interface-body* جاری، زمانی که به نماد وارد شده ارجاع می‌شود به عنوان پیشوند استفاده شود.

هر *import-symbol* باید یک *identifier* باشد که به وسیله یک *value-decl*، *type-decl*، *procedure-decl*، یا یک *termination-decl* در *interface-body* تعریف نوع واسطی که به وسیله *object-identifier* در عبارت import آورده شده است، تعریف می‌شود. فقط *import-symbol* هایی که در فهرست *import-symbol* می‌آیند، باید در محدوده‌ی *interface-body* جاری استفاده شوند. معنای همبسته با *import-symbol*، معنایی است که در تعریف نوع واسط تعریف کننده‌اش دارد. در صورتی که هیچ *import-symbol-lis* ای موجود نباشد کل واسط وارد می‌شود. این کار معادل وارد کردن صریح (با یک *import-symbol*) شناسه‌هایی است که به وسیله *value-decl* ، *type-decl*، *procedure-decl*، و *termination-decl* در تعریف نوع واسط ارجاع شده، تعریف شده‌اند.

۷-۴ اعلان‌هاي مقدار

```
value-decl = "value" value-identifier ":"
constant-type-spec "=" value-expression.
```

value-identifier = identifier.

constant-type-spec = integer-type | real-type | character-type |
boolean-type | enumerated-type | state-type |
ordinal-type | time-type | bit-type | rational-type |
scaled-type | complex-type.

value-expression = value-reference | procedure-reference |
integer-literal | real-literal | character-literal |
boolean-literal | enumerated-literal | state-literal |
ordinal-literal | time-literal | bit-literal |
rational-literal | scaled-literal | complex-literal |
void-literal.

یک *value-decl* یک *identifier* را برابر با مقدار ثابتی از نوع داده‌ی ارائه شده، اعلان می‌کند. سپس این *identifier* جایی که *value-expression* آن نوع داده در تعریف نوع واسط ممکن است (به عنوان مثال، اعلان حدود یک آرایه) به کار رود، مجاز است مورد استفاده قرار بگیرد.

همه *value-identifier*ها، باید در *interface-type* بلافاصله در بر گیرنده‌شان یکتا باشند. یک *value-expression*، یا یک *literal* (مقدار بلافصل) از نوع مشخص شده یا یک *value-reference* است. این *value-reference* باید به یک *value-expression* که در *value-decl* دیگری اعلان شده است یا به یک لفظ شمارشی[1] (در صورتی که نوع داده‌ی مشخص شده شمارشی باشد) ارجاع کند.

۷-۵ اعلان‌هاي نوع داده

پاراگراف‌هایی از این بند که برای یک تفسیر صوری به ISO/IEC 11404 ارجاع می‌دهند برای کامل بودن و کمک به خواننده آورده شده‌اند، و قسمت‌های اطلاعاتی این استاندارد ملی هستند.

type-decl = "type" type-identifier "=" type-specifier | parameterized-type-decl.

type-identifier = identifier.

یک اعلان نوع داده، اعلان می‌کند یک *identifier* از نوع مشخصی باشد. این *identifier* ممکن است سپس هرجایی که امکان دارد یک *type-specifier* در واسط استفاده شود، به کار رود (به عنوان مثال، برای تعریف نوع داده‌ی پارامتر در اعلان رویه). معنی‌شناسی و نحو *parameterized-type-decl* در بند ۷-۶ آورده شده است.

تمام *type-identifier* ها، باید در *interface-type* بلافاصله در بر گیرنده یکتا باشند. معنی‌شناسی[2] تمام انواع داده‌ای آورده شده در این سند با معنی‌شناسی انواع داده‌ای که در انواع داده‌ای مستقل از زبان – ISO/IEC 11404 تعریف شده‌اند همخوانی دارد.

type-specifier = primitive-datatype | generated-datatype | defined-datatype.

---

1 - Enumeration literal
2 - Semantics

defined-datatype = type-reference [subtype-spec].

یک *type-reference* در ساخت *defined-datatype* باید به یک *type-specifier* ارجاع کند. *type-identifier* تعریف شده در *type-decl* بلافاصله در بر گیرنده، مترادفِ *type-specifier* ای است که *defined-datatype* به آن ارجاع کرده است. اگر *type-reference* به یک *integer-type*، *real-type*، یا یک *enumerated-type* ارجاع کند، یک *subtype-spec* اختیاری نیز ممکن است داشته باشد. در صورتی که *type-reference* به *real-type* ارجاع کند و *subtype-spec* نیز داشته باشد، آن *subtype-spec* باید فقط یک *محدوده* تک از مقادیر حقیقی داشته باشد.

۷-۵-۱ انواع داده‌ی اصلی

primitive-datatype = integer-type | real-type | character-type |
boolean-type | enumerated-type | octet-type |
procedure-type | state-type | ordinal-type |
time-type | bit-type | rational-type |
scaled-type | complex-type | void-type.

۷-۵-۱-۱ عدد صحیح (integer)

عدد صحیح، نوع داده‌ی ریاضی است که شامل مقادیر دقیق بدون خرده[1] می‌شود. نحو:

integer-type = "integer".

integer-literal = ["_"]digit{digit}.

تفسیر نحو به طور صوری در ISO/IEC 11404 تعریف شده است.

۷-۵-۱-۲ نوع داده‌ی حقیقی (real)

خانواده‌ای از انواع داده‌ای است که تقریب‌های محاسباتی انواع داده‌ی ریاضی که «اعداد حقیقی» را تشکیل می‌دهند، هستند. به طور خاص، هر نوع داده‌ی حقیقی مجموعه‌ای از مقادیر حقیقی ریاضی را معین می‌کند که برای کاربردهای معین و با دقت محدود شناخته می‌شوند و باید حداقل با آن دقت در آن کاربردها قابل تمایز باشند. نحو:

real-type = "real" ["(" radix "," factor ")"].

radix = value-expression.

factor = value-expression.

real-literal = integer-literal ["."digit{digit}] [[" _ "] "E" digit{digit}].

تفسیر نحو به طور صوری در ISO/IEC 11404 تعریف شده است.

۷-۵-۱-۳ نوع داده‌ی نویسه (character)

نویسه، خانواده‌ای از انواع داده است که فضاهای مقدار آن مجموعه‌ی-نویسه‌ها است. نحو:

character-type = "character" ["(" repertoire-list")"].

---

1 - Integral

repertoire-list = repertoire-identifier {"," repertoire-identifier}.

repertoire-identifier = value-expression.

character-literal = "'"character"'".

character =

The value of character shall be any character drawn from the character set identified by the repertoire identifier in the production character-type, or from the default character set if the repertoire identifier is absent.

تفسیر نحو به طور صوری در ISO/IEC 11404 تعریف شده است.

۷-۵-۱-۴ نوع داده بولی (boolean)

بولی، نوع داده‌ی ریاضی است که منطق دو-ارزشی دارد. نحو:

boolean-type = "boolean".

boolean-literal = "true" | "false".

تفسیر نحو به طور صوری در ISO/IEC 11404 تعریف شده است.

۷-۵-۱-۵ نوع داده‌ی شمارشی (enumerated)

شمارشی، خانواده‌ای از انواع داده است که هر یک از آن‌ها از تعداد محدودی از مقادیر متمایزی که ترتیب ذاتی دارند تشکیل می‌شوند. نحو:

enumerated-type = "enumerated" "(" enumerated-value-list ")".

enumerated-value-list = enumerated-literal {"," enumerated-literal}.

enumerated-literal = identifier.

تفسیر نحو به طور صوری در ISO/IEC 11404 تعریف شده است.

۷-۵-۱-۶ نوع داده‌ی هشت‌تایی (octet)

octet-type = "octet".

بر طبق انواع داده‌ای مستقل از زبان – ISO/IEC 11404، نوع داده‌ای هشت‌تایی، نوع داده‌ی مشتق شده است: آرایه‌ای به طول ۸ از بیت‌ها: array (1..8) of (bit).

۷-۵-۱-۷ نوع داده‌ی رویه (procedure)

رویه، نوع داده‌ای تولید می‌کند (که نوع داده‌ی رویه نامیده می‌شود) که هر یک از مقادیر آن عملی بر روی مقادیر انواع داده‌ای آرگومان‌ها است. یعنی نوع داده‌ی رویه از مجموعه‌ی تمامی اعمال بر روی مقادیر یک مجموعه‌ی ویژه از انواع داده‌ای، تشکیل می‌شود. تمام مقادیر یک نوع داده‌ی رویه، تجزیه‌ناپذیر[1] هستند. نحو:

procedure-type = "procedure" "(" [argument-list] ")"

["returns" "(" return-argument ")"]

1 - Atomic

["raises" "(" termination-list ")"].

argument-list = argument-declaration {"," argument-declaration}.

argument-declaration = direction argument.

direction = "in" | "out" | "inout".

argument = argument-name ":" ["restricted"] argument-type.

argument-type = type-specifier.

argument-name = identifier.

return-argument = [argument-name ":"] argument-type.

termination-list = termination-reference {"," termination-reference}.

termination-reference = [interface-synonym "::"] identifier.

یک procedure-declaration ، به عنوان بخشی از همبستگی interface-type (به بند ۶-۹ مراجعه شود)، یک اسم را به یک procedure-type همبسته می‌کند.

termination-reference ها، باید در *interface-type* بلافاصله در بر گیرنده یکتا باشند.

۷-۱-۵-۱-۷ پارامترهای رویه

یک نوع آرگومان (*argument-type*) می‌تواند هر نوع داده‌ای را نشان دهد. نام آرگومانها[1] در *argument-list*، باید با یکدیگر و از *نام-آرگومانِ* *return-argument* (در صورت وجود) متفاوت باشند. *termination-reference* ها در *termination-list* (در صورت وجود) باید متمایز باشند.

۷-۱-۵-۲-۷ مقادیر رویه

همانطور که در بند ۶-۶ تعریف شد، مقادیر نوع-رویه، بستارهای رویه هستند. یک *argument* در *argument-list* را آرگومان ورودی می‌گوییم در صورتی که *argument-declaration* آن شامل جهت «in» یا «inout» باشد. فضای ورودی حاصل ضرب (دکارتی)[2] فضاهای مقدارِ انواع داده‌ای است که به وسیله *argument-type های* تمام آرگومان‌های ورودی تعیین می‌شود. در صورتی که آرگومانی *return-argument* باشد یا در *argument-list*، *argument-declaration* آن شامل جهت «out» یا «inout» باشد، به آن آرگومان نتیجه می‌گوییم. فضای نتیجه‌ی عادی، حاصل ضرب فضاهای مقدارِ انواع داده‌ای است که به وسیله *argument-type* های تمام آرگومان‌های نتیجه (در صورت وجود) تعیین می‌شود و در غیر این صورت حاصل ضرب فضای مقدار نوع داده‌ی تُهی[3] است. زمانی که *termination-list* وجود نداشته باشد، فضای نتیجه‌ی نوع داده‌ی رویه، فضای نتیجه‌ی عادی است، و هر مقدار p نوع داده‌ی رویه، تابعی به شکل ریاضی:

$$P: I_1*I_2*\dots I_n \rightarrow R_p*R_1*R_2*\dots R_m$$

است که در آن I_k فضای مقدار نوع داده‌ی آرگومانِ kاُمین آرگومان ورودی و R_k فضای مقدار نوع داده‌ی آرگومانِ kاُمین آرگومان نتیجه، و R_p فضای مقدار return-argument است.

---

1 - Argument-name
2 - Cross-product
3 - Void

زمانی که *termination-list* وجود داشته باشد، هر *termination-reference* به وسیله *termination-declaration* ای به یک فضای نتیجه‌ی جایگزین (که حاصل ضرب فضاهای مقدارِ انواع داده‌ای است که به وسیله *argument-type* های *argument* ها در *termination-argument-list* معین می‌شوند) مرتبط می‌شود. در صورتی که A^j فضای نتیجه‌ی جایگزینِ jاُمین پایان‌دهی باشد. بنا بر این:

$$A^j = E^j_1 * E^j_2 * \dots E^j_{mj}$$

که در آن E^j_k، فضای مقدارِ نوع داده‌ی آرگومانِ kاُمین آرگومان در *termination-argument-list* از jاُمین پایان‌دهی است. سپس فضای نتیجه‌ی عادی، فضای نتیجه‌ی جایگزینِ مربوط به پایان‌دهی عادی (A^0) که با *termination-identifier* «normal*» مدل می‌شود، خواهد بود. *termination-reference* ها، و «normal*» را برای نمایش مقادیرِ یک نوع داده‌ی حالت نامشخص S_T در نظر می‌گیریم. بنابر این، فضای نتیجه‌ی نوع داده‌ی رویه، عبارت زیر خواهد بود:

$$S_{T*}(A^0 \mid A^1 \mid A^2 \mid \dots \mid A^N),$$

که در آن A^0، فضای نتیجه‌ی عادی و A^k فضای نتیجه‌ی جایگزینِ kاُمین پایان‌دهی است، و هر مقدارِ نوع داده‌ی رویه تابعی به شکل زیر خواهد بود:

$$P: I1 * I2 * \dots * In \rightarrow S_{T*}(A^0 \mid A^1 \mid A^2 \mid \dots \mid A^N)$$

هر کدام از فضاهای ورودی، فضای نتیجه‌ی عادی و فضای نتیجه‌ی جایگزین متناظر با *termination-identifier* مفروض، ممکن است خالی باشند. یک فضای خالی می‌تواند به صورت ریاضی با جایگزین کردن فضای خالی با فضای مقدارِ نوع داده‌ی تهی، مدل شود.

فضای مقدار یک نوع داده‌ی رویه به طور مفهومی شامل تمام عملیاتی که با مدل بالا مطابقت می‌کنند، می‌شود. یعنی اعمالی که بر روی مجموعه مقادیری عمل می‌کنند که انواع داده‌ای‌شان با انواع داده‌ی آرگومان ورودی مطابقت می‌کند و منجر به مجموعه مقادیری می‌شوند که نوع داده‌شان با انواع داده‌ی آرگومانِ فضای نتیجه‌ی عادی یا فضای نتیجه‌ی جایگزینِ مناسب، مطابقت می‌کند. اصطلاح «متناظر[1]» در این‌جا به این معنی است که برای هر نوع داده‌ی آرگومان در فضای ساخت مربوطه، «مجموعه‌ی مقادیر» باید به‌طور دقیق مربوط به یک مقدار از آن نوع داده باشد. زمانی که فضای ورودی خالی است، فضای مقدار نوع داده‌ی رویه شامل تمام عملیات فاقد آرگومانی است که منجر به مقادیری در فضای نتیجه می‌شود. زمانی که فضای نتیجه خالی است، فضای مقدار ریاضی فقط شامل یک مقدار است، اما فضای مقدارِ نوع داده‌ی رویه محاسباتی می‌تواند شامل مقادیر متفاوت زیادی شود که تأثیرشان بر «جهان واقعی» (یعنی عملیات فیزیکی خارج از فضای اطلاعاتی) متفاوت است. نحوِ مقدار:

```
procedure-declaration =
   "procedure" procedure-identifier "(" [argument-list] ")"
   ["returns" "("return-argument")"]
   ["raises" "("termination-list.
procedure-identifier = identifier.
```

1 - Corresponding

procedure-reference = "procedure-identifier.

یک *procedure-declaration*، *procedure-identifier* ای را اعلان می‌کند که به یک مقدار (خاص) از نوع داده‌ی رویه‌ی که *type-specifier* آن با *procedure-declaration* پس از حذف *procedure-identifier* یکسان است، ارجاع می‌کند.

۷-۵-۱-۷-۳ زیرنوع‌های رویه

برای دو نوع داده‌ای رویه P و Q:

- P به طور صوری با Q سازگار[1] است در صورتی که: طول *argument-list* هایشان یکسان باشد، جهت هر *آرگومان* در *argument-list* P با *argument* متناظرش در *argument-list* Q یکی باشد، یا هر دو *return-argument* داشته باشند یا هیچ یک نداشته باشند، *termination-list* های P و Q (در صورت وجود) شامل *termination-reference* های یکسانی باشند.
- در صورتی که P به طور صوری با Q سازگار باشد، و برای هر آرگومان نتیجه‌ی Q، نوع داده‌ی آرگومانی که متناظر با آرگومان P است، زیرنوعی از نوع داده‌ی آرگومانِ آرگومان Q باشد، گفته می‌شود P زیرنوع-نتیجه‌ی Q است. در صورتی که نوع داده‌ی آرگومان بازگشتی و تمام انواع داده‌ی آرگومان در *argument-list* آن دو یکسان باشند، هریک زیرنوع-نتیجه‌ی دیگری است.
- در صورتی که P به طور صوری با Q سازگار باشد، و برای هر آرگومان ورودی Q، نوع داده‌ی آرگومانی که متناظر با آرگومان P است، زیرنوعی از نوع داده‌ی آرگومانِ آرگومان Q باشد، گفته می‌شود P زیرنوع-ورودی Q است. در صورتی که تمام انواع داده‌ی آرگومان ورودی در *argument-list* آن دو یکسان باشند، هریک زیرنوع-ورودی دیگری است.

۷-۵-۱-۸ نوع داده‌ی حالت (state)

حالت، خانواده‌ای از انواع داده‌ای است که هریک از آن‌ها تعداد محدودی از مقادیر متمایز و بدون ترتیب را در بر می‌گیرد و هیچ‌گونه عملیات توصیفی به غیر از مساوی (equal) ندارد. نحو:

state-type = "state" "(" state-value-list ")".

state-value-list = state-literal {"," state-literal}.

state-literal = identifier.

تفسیر نحو به طور صوری در ISO/IEC 11404 تعریف شده است.

۷-۵-۱-۹ نوع داده‌ی وصفی (ordinal)

وصفی، نوع داده‌ی اعداد وصفی است که متفاوت از اعداد کمّی (نوع داده‌ی عدد صحیح) هستند. وصفی، نوع داده‌ی شمارشی نامحدود است. نحو:

ordinal-type = "ordinal".

ordinal-literal = digit {digit}.

تفسیر نحو به طور صوری در ISO/IEC 11404 تعریف شده است.

1 - Compatible

۷-۵-۱-۱۰ نوع داده‌ی زمان (time)

زمان، خانواده‌ای از انواع داده‌ای است که مقادیر آن، نقاطی در زمان با دقت‌های رایج مختلف (سال، ماه، روز، ساعت، دقیقه، ثانیه، و کسرهای آن) است. نحو:

time-type = "time" "(" time-unit ["," radix "," factor]")".

time-unit = "year" | "month" | "day" | "hour" | "minute" | "second" |

parametric-value.

time-literal = digit{digit} ["."digit{digit}].

تفسیر نحو به طور صوری در ISO/IEC 11404 تعریف شده است.

۷-۵-۱-۱۱ نوع داده‌ی بیت (bit)

بیت، نوع داده‌ای است که فیلد محدودِ دو نمادی که «0» (همانی جمع) و «1» (همانی ضرب) را نشان می‌دهند را نشان می‌دهد. نحو:

bit-type = "bit".

bit-literal = "0" | "1".

تفسیر نحو به طور صوری در ISO/IEC 11404 تعریف شده است.

۷-۵-۱-۱۲ نوع داده‌ی گویا (rational)

گویا، نوع داده‌ی ریاضی است که شامل «اعداد گویا» می‌شود. نحو:

rational-type = "rational".

rational-literal = ["_"] digit{digit} ["/" digit{digit}].

تفسیر نحو به طور صوری در ISO/IEC 11404 تعریف شده است.

۷-۵-۱-۱۳ نوع داده‌ی مختلط (complex)

خانواده‌ای از انواع داده‌ای است که فضاهای مقدار آن، زیرنوعی از فضای مقدار گویا است، اما انواع داده‌ای مقیاس‌دار، مقدار تقریبی دارند. نحو:

scaled-type = "scaled" "(" radix "," factor ")".

scaled-literal = ["_"] digit{digit} [fraction].

fraction = "." digi t{digi t}.

تفسیر نحو به طور صوری در ISO/IEC 11404 تعریف شده است.

۷-۵-۱-۱۴ نوع داده‌ی مختلط (complex)

مختلط، خانواده‌ای از انواع داده است که هر یک تقریب محاسباتی نوع داده‌ی ریاضی «اعداد مختلط» هستند. هر نوع داده‌ی مختلط، یک مجموعه از مقادیر مختلط ریاضی (که با یک دقت محدود برای برخی کاربردها شناخته شده‌اند و باید حداقل با آن دقت در آن کاربردها قابل تمایز باشند) را معین می‌کند. نحو:

complex-type = "complex" ["(" radix "," factor ")"].

complex-literal = "(" real-part "," imaginary-part ")".

real-part = real-literal.

imaginary-part = real-literal.

تفسیر نحو به طور صوری در ISO/IEC 11404 تعریف شده است.

۷-۵-۱-۱۴-۱ نوع داده‌ی تهی (void)

نوع داده‌ی تهی، شیءای را نشان می‌دهد که وجود آن به صورت نحوی یا معنایی الزامی است، اما نمونه‌ی مورد نظر اطلاعاتی را در بر ندارد. نحو:

void-type = "void".

void-literal = "nil".

تفسیر نحو به طور صوری در ISO/IEC 11404 تعریف شده است.

۷-۵-۲ انواع داده‌ی تولید شده (generated)

generated-datatype = record-type

| choice-type

| array-type

| pointer-type.

۷-۵-۲-۱ نوع داده‌ی رکورد (record)

رکورد، نوع داده‌ای تولید می‌کند که مقادیر آن، انبوهشی‌های ناهمگونِ مقادیرِ انواع داده‌ی مؤلفه هستند، هر انبوهشی، مقداری برای هر نوع داده‌ی مؤلفه دارد که با یک «field-identifier» ثابت کلیددهی[1] می‌شود. نحو:

record-type = "record" "of" "("field-list")".

field-list = field {"," field}.

field = field-identifier ":" field-type.

field-identifier = identifier.

field-type = type-specifier.

field-name باید در *record-type* یا *choice-type* بلافاصله در بر گیرنده یکتا باشد.

تفسیر نحو به طور صوری در ISO/IEC 11404 تعریف شده است.

1 - Keyed

۷-۵-۲-۲ نوع داده‌ی انتخاب (choice)

انتخاب، نوع داده‌ای تولید می‌کند، که هر یک از مقادیر آن، یک مقدار تک از یکی از مجموعه‌ی انواع داده‌ای جایگزین است. انواع داده‌ی جایگزینِ مربوط به یک نوع داده‌ی انتخاب به صورت منطقی به وسیله تناظرشان به مقادیرِ نوع داده‌ی دیگر (نوع داده‌ی tag) متمایز می‌شوند.

choice-type = "choice" "("tag-type")" "of" "("alternative-list")".

tag-type = type-specifier.

alternative-list = alternative { "," alternative} [default-alternative].

alternative = tag-value-list ":" alternative-type.

default-alternative = "default" ":" alternative-type.

alternative-type = type-specifier.

tag-value-list = select-list.

select-list = select-item {"," select-item}.

select-item = value-expression | select-range.

select-range = lowerbound ".." upperbound.

lowerbound = value-expression | "*" .

upperbound = value-expression | "*" .

تفسیر نحو به طور صوری در ISO/IEC 11404 تعریف شده است.

۷-۵-۲-۳ نوع داده‌ی آرایه (array)

آرایه، نوع داده‌ای تولید می‌کند که مقادیر آن بین فضای ساخت یکی یا بیشتر از یکی از انواع داده‌ای محدود (نوع داده‌ی شاخص[1]) و فضای مقدارِ نوع داده‌ی عنصر[2] قرار دارد. به نحوی که هر مقدار در فضای ساختِ انواع داده‌ای شاخص، به‌طور دقیق به یک مقدار از نوع داده‌ی عنصر مربوط می‌شود. نحو:

array-type = "array" "("index-type-list")" "of" "("element-type")".

index-type-list = index-type {"," index-type}.

index-type = type-specifier | index-lowerbound".."index-upperbound.

index-lowerbound = value-expression.

index-upperbound = value-expression.

element-type = type-specifier.

تفسیر نحو به طور صوری در ISO/IEC 11404 تعریف شده است.

1 - Index
2 - Element

۷-۵-۲-۴ نوع داده‌ی اشاره‌گر (pointer)

اشاره‌گر، نوع داده‌ای تولید می‌کند که هر یک از مقادیر آن به منزله‌ی ابزار ارجاع به مقادیرِ نوع داده‌ی دیگر (نوع داده‌ی عنصر) است. مقادیر نوع داده‌ی اشاره‌گر تجزیه ناپذیر هستند. نحو:

pointer-type = "pointer" "to" "("element-type")".

اشاره‌گر، مولد نوعی است که نوع داده‌ی اصلی تولید می‌کند که هریک از مقادیر آن به منزله‌ی ابزار ارجاع به مقادیرِ نوع داده‌ی دیگر (*element-type*) است. همانطور که در بند ۶-۷ تعریف شد، مقادیر نوع داده‌ی اشاره‌گر، جعبه‌ها هستند.

یک اشاره‌گر با صفت «محدود شده»[1]، اشاره‌گری است که هرگز برچسب تهی (پوچ)[2] ندارد، و نه به صورت ایستا و نه به صورت پویا، مستعار[3] با اشاره‌گر دیگری نمی‌شود. اشاره‌گرهای محدود شده می‌توانند به صورت کارا پشتیبانی شوند. اگرچه به دلیل پروتکل بهینه‌شده[4]، تعیین این‌که آیا برچسب یک اشاره‌گر محدودشده‌ی ورودی-خروجی (inout) در نتیجه‌ی اجرای رویه کارساز تغییر کرده است یا خیر، امری غیر ممکن است.

در صورتی که بیش از یک جعبه، مقدار اشاره‌گر را بین همبسته‌های تعمیم یافته‌ی همبستگی احضار در هنگام راه‌اندازی داشته باشند، به مقدار آن اشاره‌گر به صورت ایستا در احضار رویه نام مستعار داده شده است.

یادآوری - مستعار کردن ایستا، خاصیتی از بستار است، در حالی که مستعار کردن پویا خاصیتِ احضار است. در تعاریف بالا فرض می‌شود که یک پارامتر صوری در همبستگی احضاری که حاوی مقدار پارامتر واقعی است، تبدیل به یک جعبه می‌شود. با این‌که به‌طور واقعی نیاز نیست اما، نمادگذاری جعبه باید برای در بر گرفتن اتصال‌های پارامتر صوری/واقعی، بسط داده شود (فقط برای اهداف تعریف بالا).

۷-۵-۳ زیرنوع‌ها

subtype-spec = "select" "("select-element {"," select-element}")".

select-element = value-expression | range.

range = lower-bound ".." upper-bound | ".." upper-bound | lower-bound ".."

lower-bound = value-expression.

upper-bound = value-expression.

یک *subtype-spec* شامل فهرستی از عناصر می‌شود، که در آن هر عنصر یا *value-expression* نوع داده‌ی مشخص شده است یا *محدوده‌ای* از مقادیرِ نوع مشخص شده است. *value-expression* ای که در یک *subtype-spec* می‌آید، باید یا به یک لفظ (مقدار بلافصل)، یا به یک لفظ شمارشی، یا به یک *formal-value-parm* ارجاع کند.

۷-۶ انواع پارامتري شده (parameterized)

parameterized-type-decl =

1 - Restricted
2 - Null label
3 - Alias
4 - Optimized protocol

"type" type-identifier "("formal-value-parms")" "=" type-specifier.

formal-value-parms = formal-value-parm {"," formal-value-parm}.

formal-value-parm = identifier ":" value-param-type-spec.

value-parm-type-spec = type-specifier.

parameterized-type-decl، مشخصات ناقص یک نوع داده را نشان می‌دهد. *parameterized-type-decl*، یک *type-identifier* و یک مجموعه از پارامترهای صوری (که *formal-value-parm* نامیده می‌شوند) را به *type-specifier* همبسته می‌کند. هر *formal-value-parm*، خودش *identifier* ای است که می‌توان از *type-specifier* به آن ارجاع کرد. ارجاع به این *formal-value-parm*، فقط می‌تواند در مکان *value-expression* از *type-specifier* صورت بگیرد (برای مثال در مکانِ حدود یک آرایه).

هر *formal-value-parm* یک *value-param-type-spec* دارد که نوع داده‌ی *formal-value-parm* را مشخص می‌کند. این نوع، باید نوع داده‌ای باشد که *value-expression* ممکن است در یک تعریف نوع واسط داشته باشد.

type-identifier آورده شده در *parameterized-type-decl* می‌تواند (به شرطی که مقادیر واقعی برای *formal-value-parm* های *parameterized-type-spec* تأمین شده باشد) هر جایی که یک *type-specifier* در واسط می‌تواند به کار برود، استفاده شود. بنابر این زمانی که به این *type-identifier* ارجاع می‌شود، باید به عنوان *parameterized-type-reference* مورد ارجاع قرار بگیرد.

یک *parameterized-type-decl* نباید به‌طور مستقیم به خودش ارجاع کند (به وسیله *parameterized-type-reference*). همچنین نباید به خودش به صورت غیر مستقیم نیز ارجاع کند (به وسیله *parameterized-type-reference* به یک نوع پارامتری شده‌ی متفاوت که به صورت مستقیم یا غیر مستقیم به این نوع پارامتری شده ارجاع می‌کنند).

همه *formal-value-parm* ها باید در *parameterized-type-decl* بلافصل‌شان یکتا باشند.

۷-۷ شناسه‌ها

object-identifier = "{"ObjectIdComponent {ObjectIdComponent}"}".

ObjectIdComponent = identifier | digit | identifier "("digit {digit}")".

نحوِ *object-identifier*، نحوِ یک ASN.1 ObjectIdentifierValue (همان‌طور که در ISO 8824 تعریف شده) است.

type-reference = [interface-synonym "::"] identifier |

parameterized-type-reference.

parameterized-type-reference = [interface-synonym "::"]

identifier "("actual-value-parm

{"," actual-value-parm}")".

actual-value-parm = value-reference.

هر زمان که در یک *interface-type* از *parameterized-type-reference* استفاده شود، باید به *type-specifier* از *parameterized-type-decl* ارجاع کند. برای هر *formal-value-parm* از *parameterized-type-decl*، *باید* یک *actual-value-parm* ارائه شود. نوع داده‌ی یک *actual-value-parm* باید با نوع داده‌ی *formal-value-parm* متناظرش یکسان باشد. معنی‌شناسی *type-specifier* به‌دست‌آمده، آن است که با جایگزین کردن ارجاع‌های *formal-value-parm* در *type-specifier* با *actual-value-parm* متناظرشان، حاصل شده است.

value-reference = [interface-synonym "::"] identifier {"." Identifier}.

identifier = letter {pseudo-letter}.

letter = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" |
"P" |" Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z" |
"a" | "b" |"c" | "d" | "e" |"f" | "g" | "h" | "i" | "j" | "k" | "1" | "m" | "n" | "o" |
"p" | "q" | "r" | "s" | "t" |"u" | "v" | "w" |"x" | "y" | "z" .

pseudo-letter = letter | digit | underline.

digi t = "0" | "1" |" 2" | "3" |" 4" |" 5" |" 6" |" 7" |" 8" |" 9" .

underline = "_"

۷-۷-۱ ارجاع‌های مقدار به فیلدها

یک *value-reference* با یک *field* در صورتی مطابقت دارد که:

۱- *field* به صورت بلافصل در یک *record-type*، R قرار داشته باشد، و

۲- *value-reference* در R قرار داشته باشد، و

۳- اولین مؤلفه‌ی شناسه‌ی *value-reference* با *field-name* *فیلد* یکسان باشد، و

۴- *value-reference* به یک *procedure-type* که در R قرار دارد، محدود نشده باشد، و

۵- یک *record-type* مانند R2 در R وجود نداشته باشد به نحوی که زمانی که R2 به جای R جایگزین می‌شود، موارد ۱ تا ۴ برقرار باشند.

در صورتی که *value-reference* با یک *field* مطابقت کند، اولین *identifier* از *value-reference* به آن *field* ارجاع می‌کند. در صورتی که iاُمین *identifier* مربوط به *value-reference* به یک *field* ارجاع کند و *value-reference* بیش از i شناسه داشته باشد، *field* ای که شناسه‌ی iاُم به آن ارجاع می‌کند، باید یک *record-type*، و i+1اُمین شناسه‌ی *value-reference* باید با *field-name* این *record-type* یکسان باشد. i+1اُمین شناسه‌ی *value-reference* به یک *field* ارجاع می‌کند که به این *field-name* همبسته است. در صورتی که iاُمین شناسه‌ی *value-reference* به یک *field* ارجاع کند و *value-reference* به‌طور دقیق i شناسه داشته باشد، *value-reference* به این *field* ارجاع می‌کند.

۷-۷-۲ ارجاع‌های مقدار به پارامترها، آرگومان‌های-بازگشتی[1]، یا فیلدهایی که در آن قرار دارند

یک *value-reference* با یک *parameter* (*return-arg*) در صورتی مطابقت دارد که:

۱- *value-reference* با *field* ای مطابقت نکند، و

۲- *parameter (return-arg)* به صورت بلافصل در *procedure-decl* یا *procedure-type* P قرار داشته باشد، و

۳- *value-reference* در P قرار داشته باشد، و

۴- اولین مؤلفه‌ی شناسه‌ی *value-reference*، با *parameter-name* (*identifier*) مربوط به *parameter* (*return-arg*) یکسان باشد، و

۵- *value-reference* در یک *procedure-type* (به غیر از P) که در P قرار دارد، قرار نداشته باشد.

۶- در صورتی که *value-reference* با یک *parameter* (*return-arg*) مطابقت کند و *value-reference* شامل یک تک *identifier* باشد، *value-reference* به آن *parameter* (*return-arg*) ارجاع می‌کند. در غیر این صورت، *parameter* (*return-arg*) باید یک *record-type* باشد و *value-reference* باید به یک *field* ارجاع کند (مطابق قواعد بند ۷-۷-۱).

۷-۷-۳ ارجاع‌های مقدار به formal-value-parms

یک *value-reference* در صورتی با یک *formal-value-parm* مطابقت می‌کند که:

۱- *value-reference* با یک *field*، یک *parameter*، یا یک *return-arg* مطابقت نکند، و

۲- *formal-value-parm* در یک *parameterized-type-decl* به صورت بلافصل قرار داشته باشد، و

۳- *value-reference* در *type-specifier* مربوط به این *parameterized-type-decl* قرار داشته باشد و با *formal-value-parm* یکسان باشد.

اگر یک *value-reference* با یک *formal-value-parm* مطابقت کند، آنگاه به آن *formal-value-parm* ارجاع می‌کند.

۷-۷-۴ ارجاع‌های مقدار به value-expressions

value-reference در صورتی با یک *value-decl* مطابقت می‌کند که *value-identifier* از *value-decl*، با مؤلفه‌ی *identifier* از *value-reference* یکسان باشد.

اگر مؤلفه‌ی *interface-synonym* از *value-reference* وجود نداشته باشد، و *value-reference* با *value-decl* ای در *interface-type* بلافاصله در بر گیرنده مطابقت کند، و *value-reference* با یک *field*، یک *parameter*، یک *return-arg*، یا یک *formal-value-parm* مطابقت نکند، *value-reference* به *value-expression* بلافصل آن *value-expression* ارجاع می‌کند. در غیر این صورت، در صورتی که مؤلفه‌ی *interface-synonym* از *type-reference* وجود نداشته باشد، و *value-reference* به‌طور دقیق با یک *value-decl* وارد شده مطابقت کند، و *value-reference* با یک *field*، یک *parameter*، یک *return-arg*، یا

1 - Return-args

یک *formal-value-parm* مطابقت نکند، *value-reference* به *value-expression* بلافاصله قرار گرفته‌ی آن *value-decl*، ارجاع می‌کند.

یادآوری – در صورتی که *value-identifier* یک *value-decl* وارد شده، همانند یک *value-identifier* باشد که در *interface-type* بلافاصله در بر گیرنده تعریف شده است، یا با *value-identifier* از یک *value-decl* که از تعریف نوع واسط متفاوتی وارد شده است، یکسان باشد، فقط می‌تواند با استفاده از *interface-synonym* مربوطه‌اش مورد ارجاع قرار بگیرد.

در صورتی که مؤلفه‌ی *interface-synonym* از *value-reference* وجود داشته باشد و *value-reference* با *value-decl* در تعریف نوع واسطی که با *interface-synonym* مشخص شده، مطابقت کند، *value-reference* به *value-expression* بلافصلِ قرار گرفته مربوط به این *value-decl*، ارجاع می‌کند.

۷-۷-۵ ارجاع‌های مقدار به enumeration-identifiers

زمانی که مؤلفه‌ی *type-identifier* از *value-reference* وجود داشته باشد، یک *value-reference* با *enumeration-identifier* از یک *enumerated-type* در صورتی مطابقت می‌کند که *type-identifier* از *value-reference*، با *enumeration-identifier* از *enumerated-type* یکسان باشد. در صورتی که *tyep-identifier* وجود نداشته باشد، یک *value-reference* در صورتی با *enumeration-identifier* از یک *enumerated-type* مطابقت می‌کند که مؤلفه‌ی *identifier* از *value-reference* با *enumeration-identifier* از *enumerated-type* یکسان باشد.

در صورتی که مؤلفه‌ی *interface-synonym* از *value-reference* وجود نداشته باشد، و *value-reference* به‌طور دقیق با یک *enumeration-identifier* در *interface-type* بلافاصله در بر گیرنده، مطابقت کند، و *value-reference* با یک *field*، یک *parameter*، یک *return-arg*، یک *formal-value-parm*، یا یک *value-expression* مطابقت نکند، *value-reference* به *enumeration-identifier* ای که با آن مطابقت دارد، ارجاع می‌کند. در غیر این صورت، در صورتی که مؤلفه‌ی *interface-synonym* از *value-reference* وجود نداشته باشد، و *value-reference* به‌طور دقیق با یک *enumeration-identifier* وارد شده مطابقت کند، و *value-reference* با یک *field*، یک *parameter*، یک *return-arg*، یک *formal-value-parm*، یا یک *value-expression* مطابقت نکند، *value-reference* به *enumeration-identifier* وارد شده‌ای که با آن مطابقت دارد، ارجاع می‌کند.

در صورتی که مؤلفه‌ی *interface-synonym* از *value-reference* وجود داشته باشد، و *value-reference* به‌طور دقیق با یک *enumeration-identifier* در تعریف نوع واسطِ مشخص شده به وسیله *interface-synonym* مطابقت کند و *value-reference* با یک *value-expression* در تعریف مشخص شده به وسیله *interface-synonym* مطابقت نکند، *value-reference* به *enumeration-identifier* ای که با آن مطابقت دارد، ارجاع می‌کند.

۷-۷-۶ ارجاع‌های پایان‌دهی

قوانین حاکم بر *termination-reference*ها، همان قوانین حاکم بر *type-reference*ها است.

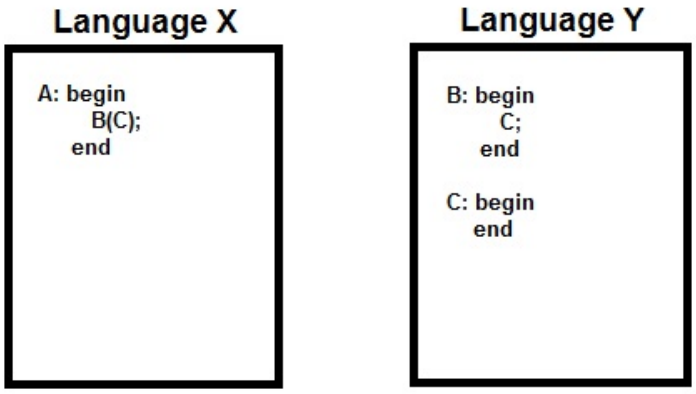
پيوست الف
(اطلاعاتي)
پارامترهاي از نوع رويه

نحوِ مربوط به ساز و کار فراخوانی مستقل‌از-زبان به یک رویه اجازه می‌دهد که پارامتری از رویه دیگر باشد. سه مورد متفاوت ناشی از ویژگی پارامترهای از نوع رویه داریم.

الف-۱ ارجاع LIPC-دسترسي محلي

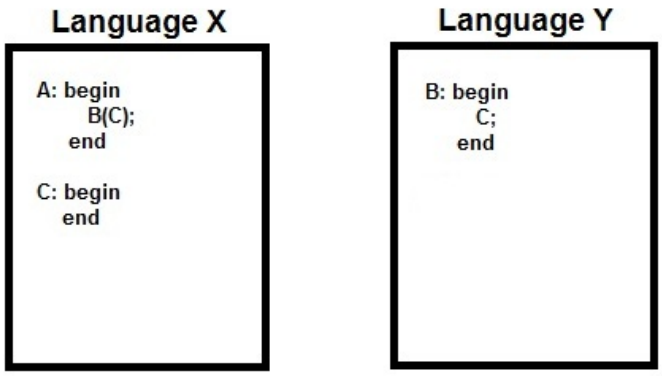
در این مورد، رویه A به زبان X، رویه B به زبان Y را فراخوانی می‌کند و به رویه B، اشاره‌گری به رویه C را که به زبان Y است ارسال می‌کند. باید روشی برای زبان X جهت ارجاع به رویه C وجود داشته باشد تا بتواند اشاره‌گری را برای ارسال به رویه B ایجاد کند. این ارجاع به C، باید به وسیله lipc-reference صورت گیرد. پس از این‌که B شروع به اجرا کرد، سرانجام C را فراخوانی می‌کند، اما این فراخوانی، یک فراخوانی محلی است و بنابر این نیازی به lipc-access نیست.

یادآوری – رویه B، باید به وسیله اطلاعاتِ آن lipc-reference که به آن ارسال شده، بفهمد چگونه رویه C را به صورت «محلی» فراخوانی کند.



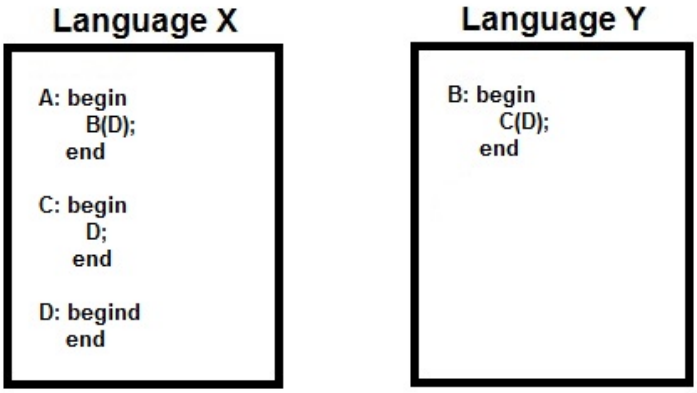
الف-۲ ارجاع LIPC-دسترسي LIPC

در این مورد، رویه A به زبان X، رویه B به زبان Y را فراخوانی می‌کند و به رویه B، یک اشاره‌گر به رویه C که به زبان X است را می‌فرستد. هنگامی که B، رویه C را فراخوانی کند باید از lipc-access استفاده کند، زیرا این فراخوانی از مرز رویه عبور می‌کند. علاوه بر این، برای این‌که B بتواند C را فراخوانی کند، باید lipc-reference رویه C را داشته باشد. این اطلاعات از اطلاعاتی که از سمت رویه A به رویه B ارسال شده است، به دست می‌آید.



الف-۳ ارجاع محلي-دسترسي محلي

در این مورد، رویه A به زبان X، رویه B در زبان Y را فراخوانی می‌کند، و اشاره‌گری به رویه‌ی D در زبان X را به رویه B می‌فرستد. در ادامه، B، رویه C که در زبان X است را فراخوانی کرده و به آن اشاره‌گری به رویه‌ی D را ارسال می‌کند. سپس، C رویه D را فراخوانی می‌کند، اما در این‌جا، هم دسترسی و هم ارجاع به D به وسیله C، محلی هستند. لذا نیازی نیست که اطلاعات اشاره‌گر D، یک lipc-reference باشد، اما باید به شکلی باشد که امکان تغییر شکل به محیط B و بازگشت به حالت اصلی خودش را بدهد.



پیوست ب
(اطلاعاتي)
نحو نشانه‌گذاري تعريف واسط

این پیوست شامل نحوِ کامل IDN، فقط جهت ارجاع به آن است.

ساخت‌های IDN **صفحه**

A

B

C

D

E

F

G

I

L

O

P

R

S

T

U

V

پیوست پ
(اطلاعاتي)
چگونگي انقياد يك زبان به LIPC

مدل LIPC، بر پایه‌ی مفهوم آشنای «کارخواه-کارساز» است: برنامه‌ی کارخواهی که رویه کارسازی را فراخوانی می‌کند. یک «قرارداد مجازی» بین دو طرف وجود دارد که در آن رویه سمت کارساز توافق می‌کند که خدمت (اجرای رویه) را ارائه کند، و برنامه‌ی سمت کارخواه نیز توافق می‌کند که اطلاعات فراخوانی ضروری (یعنی پارامترهای واقعی) را منطبق بر روش‌های ارسال پارامتری که نیاز است، تأمین کند.

بنابر این، انقیاد یک زبان برنامه نویسی به LIPC شامل دو قسمت می‌شود: یکی، مشخص کردن انقیاد هنگامی که برنامه در زبان در طریقه کارخواه عمل می‌کند و رویه LIPC ای را فراخوانی می‌کند، دیگری مشخص کردن انقیاد زمانی که برنامه‌ی نوشته شده، مقصد فراخوانی LIPC است. این انقیادها (به ترتیب) الزاماتِ خدمت LIPC کارخواه و قرارداد مجازی سمت کارخواه، و خدمت LIPC کارساز و قرارداد مجازی سمت کارساز نامیده می‌شوند. از آن‌جایی که یک پردازنده ممکن است فقط یکی از این خدمات LIPC را داشته باشد، این انقیادها باید واحدهای مجزا و جامعی باشند. یعنی پردازنده ممکن است بتواند فقط به عنوان یک کارخواه و نه به عنوان یک کارساز، یا فقط به عنوان یک کارساز و نه به عنوان یک کارخواه، یا به عنوان هم کارخواه و هم کارساز، عمل کند. در هر صورت، انقیادها باید سازگار باشند، یک فراخوانی LIPC به رویه نوشته شده در زبان یکسان، باید از فراخوانی مستقیم، غیر قابل تمایز باشد. این امر فقط به انواع وابسته-به-پردازنده‌ای که استاندارد زبان می‌پذیرد، و محدودیت‌های پیاده‌سازی اعمال شده به وسیله یک خدمت LIPC ویژه، موکول می‌شود.

نکته‌ای که باید هنگام مشخص کردن انقیادها در نظر داشت، این است که زبان‌ها ممکن است با دید بسیار مجتمع‌تری به رویه و فراخوانی آن نسبت به آن‌چه که در محیط LIPC، امکان پذیر است (یا حتی مطلوب است)، طراحی و استفاده شوند. این امر می‌تواند در استاندارد زبان منعکس شود، و باید در مواردی که زبان فرضیات ذاتی دارد که به‌طور صریح ذکر نشده‌اند، تحقیق شود که فراخوانی یک رویه به چه معنایی است. ممکن است نیاز باشد که جداسازی سمت کارخواه و سمت کارسازِ یک فراخوانی رویه، نسبت به آن‌چه که در استاندارد زبان است، کامل‌تر باشد. از این رو، جنبه‌هایی که ممکن است پوشیده باشند، باید در استانداردِ انقیاد نسبت به آن‌چه که تا کنون در جامعه‌ی زبان مرسوم بوده، صریح‌تر شوند.

بنابر این، جداسازی باید با جستجوی دقیقِ چنین فرضیات ضمنی همراه شود. زمانی که این صریح‌سازی‌ها در استاندارد انقیاد انجام می‌شود باید دقت کافی را به عمل آورد تا این فرآیند برای دیدِ کاربران زبان از فراخوانی رویه، کماکان آشنا باقی بماند.

پ-۱ پیوند دادن کارخواه و کارساز

زبان‌ها، از منظر روشی که پردازنده زبان جهت تشخیص و مکان‌یابی رویه فراخوانی شده به وسیله برنامه استفاده می‌کند، بسیار متفاوت هستند. زبان‌هایی با ساختار بلاکی سخت‌گیرانه[1]، ممکن است نیاز به «اعلان

1 - Strictly block-structured

قبل از استفاده» داشته باشند، یا حداقل آن رویه باید در بلاک یکسان، یا در بلاک‌هایی که در بر گیرنده‌ی بلاکی هستند که فراخوانی از آن‌جا آغاز شده است، تعریف شده باشد. زبان‌هایی که ساختار نامربوط[1] بیشتری دارند، که جهت ترجمه‌ی جداگانه‌ی رویه‌ها طراحی شده‌اند، ممکن است برای ایجاد ارتباطات، فرض کنند «ویراستار پیوندی»[2] وجود دارد که تعریف نشده باقی مانده یا وابسته-به-پیاده‌سازی در استاندارد زبان است. برخی از زبان‌ها نیاز به احضار صریحِ کتابخانه‌ها[3] یا پودمان[4]های مورد نیاز، یا در متن برنامه، یا به وسیله استفاده از راهنماهای[5] پردازنده خارج از خود برنامه، دارند. برخی نیز سرآیند رویه را از بدنه‌ی رویه، قابل تفکیک می‌کنند، تا بدین وسیله مشخصات پارامترهای صوری و غیره بتواند به‌طور صریح در متن برنامه‌ی فراخواننده (کارخواه) ظاهر شود.

در مورد ساختار بلاکی، رویه‌های خارجی با فرض وجود یک «اَبَر-بلاک»[6] که بیرونی‌ترین بلاک برنامه را احاطه کرده و در آن تمام رویه‌های مورد نیاز اعلان شده‌اند، می‌توانند ارائه شوند. در مورد ساختار غیر اشتراکی، پیدا کردن بلاک مفقود بر عهده‌ی ویراستار پیوند (وابسته-به-پیاده‌سازی) است، برای مثال با استفاده از دستورات پیش‌پردازنده[7] یا راهنماهای مترجم.

چنین موضوعاتی به‌طور معمول خارج از حوزه و دامنه‌ی کاربرد استاندارد زبان تلقی می‌شوند. در این مورد، استاندارد زبان باید در صورتی که شامل فرضیات ذاتی باشد، مورد بررسی قرار گیرد، اما آن‌ها می‌توانند خارج از استاندارد انقیاد زبان نیز باشند. در این صورت باید به‌طور صریح (همراه با دلیل آن) ذکر شود که آن‌ها بیرون از استاندارد انقیاد زبان هستند. فقط زمانی که استاندارد زبان به‌طور صریح در رابطه با دسترسی به پودمان‌ها یا کتابخانه‌های رویه سخن می‌گوید، ممکن است نیاز باشد که در استاندارد انقیاد، حرفی از دسترسی به رویه‌های LIPC نیز زده شود.

به طور کلی، اصل اساسی این است که فراخوانی یک رویه LIPC از درون برنامه‌ی کارخواه، باید در متن برنامه از فراخوانی یک رویه (خارجی) با زبان محلی، غیر قابل تمایز باشد. کاربرِ برنامه‌ی کارخواه باید بداند چگونه به رویه مورد نیاز دسترسی پیدا کند، اما این مورد برای هر رویه خارجی نیز صدق می‌کند. مورد اضافی دیگری نباید برای یک رویه LIPC نیاز باشد.

اگر استاندارد زبان از مدیریت استثناء[8] سخن گفته است، صرف‌نظر از نحوه‌ی برخورد زبان با رویه‌های خارجی، استاندارد انقیاد باید استثناءهای ویژه فراخوانی‌های LIPC را پوشش دهد (به عنوان مثال، عدم مکان‌یابی رویه خارجی، عدم وجود انقیادی و غیره برای پارامتر نوع داده‌ای). در بر گرفتن چنین گزارش استثنائی، حتی زمانی که استاندارد زبان خودش از استثناءها سخن نگفته باشد، ارزش ملاحظه کردن را دارد.

پ-۲ انقیاد طریقه کارخواه

1 - Disjoint structure
2 - Link editor
3 - Libraries
4 - Module
5 - Directives
6 - Super-block
7 - Pre-processor
8 - Exception

در پردازنده‌ی زبانی که در طریقه کارخواه عمل می‌کند، خدمت LIPC کارخواه ابتدا باید پارامترهای واقعی (که شامل نتیجه‌ی بازگشتی که به عنوان پارامتر «خروجی» در نظر گرفته می‌شود، هم می‌شود) را به شکل LIPC مرتب کند. مرتب کردن برای هر پارامتر واقعی هم شامل تشخیص روش ارسال پارامتر، و هم شامل نگاشت نوع داده‌ی محلی به نوع داده‌ی LID متناظر می‌شود. در استاندارد انقیاد، نیازی نیست که نگاشت‌های نوع داده ارائه شود، و می‌تواند به وسیله ارجاع به انقیاد LID زبان انجام گیرد. اما باید به‌طور صریح به هر محدودیتی که زبان بر انواع داده‌ای پارامترها اعمال می‌کند (که شامل نتیجه‌ی بازگشتی هم می‌شود)، استناد کند.

یادآوری – ممکن است شامل کردن «بسط‌های مجاز[1]» انواع داده‌ای پذیرفته شده برای پارامترها، جهت گسترده کردن محدوده‌ی رویه‌های LIPC ای که می‌توانند فراخوانی شوند، مطلوب به نظر برسد. برای مثال، موردی که انبوهشی‌ها بتوانند به صورت پارامتر ارسال شوند. که در این صورت پذیرفتن بسط‌ها برای طریقه کارساز نیز منطقی خواهد بود؛ در هر دو مورد بهتر است که بسط‌های پارامتر به جای این‌که به زمینه‌ی LIPC منحصر شوند، در وصله‌ی[2] جداگانه‌ای (اختیاری یا اجباری) در استاندارد زبان قرار داشته باشند.

به همین شکل برای انواع داده‌ای پارامترهای واقعی، استاندارد انقیاد، باید به سازوکار ارسال پارمتری که در استاندارد زبان پشتیبانی می‌شود، استناد کند و آن‌ها را به حالت‌های ارسال پارامتر در استاندارد LIPC ربط دهد. همچنین باید به قوانین مرتب کردن پارامترها از هر نوعی که باشند نیز استناد کند. در هر مورد، بایستی تصریح شود که این سازوکار: مجاز است، توصیه شده است یا الزامی است. در انقیاد LIPC، سخت‌گیر بودن نسبت به حالت محلی محض، برای بالا بردن کارایی ممکن است مطلوب باشد: به طور مثال، استاندارد زبانی ممکن است در حالت کلی استفاده از «فراخوانی با مرجع» و «رونوشت ورودی رونوشت خروجی» را برای یک پارامتر خروجی، مجاز بداند، اما برای یک فراخوانی LIPC، انقیاد می‌تواند به استفاده از «رونوشت ورودی رونوشت خروجی» برای آن توصیه کند و یا حتی آن را الزامی کند (با فرض این‌که خود رویه کارساز، این سازوکار را در قرارداد مجازی می‌پذیرد).
انقیاد طریقه کارخواه همچنین باید اجازه‌ی نامرتب کردن معکوسِ نتایج بازگشتی (یا پارامترهای خروجی) حاصل از فراحوانی را نیز بدهد.

پ-۳ انقیاد طریقه کارساز

ممکن است انقیاد طریقه کارساز، مانند تصویری در آینه، متقارن با انقیاد طریقه کارخواه به نظر برسد. اما، این به‌طور کامل درست نیست و این تقارن ظاهری، گمراه کننده است. این سمت کارساز است که قرارداد مجازی را تعیین می‌کند؛ تعریف رویه کارساز، تعداد و انواع داده‌ای پارامترها و مقدار بازگشتی (در صورت وجود) را مشخص می‌کند. همچنین سمت کارساز روش‌های مجاز ارسال پارامتر را معین می‌کند: در صورتی که کارخواه نتواند پارامتر را به روش مورد نیاز ارسال کند، فراخوانی انجام نمی‌شود.

1 - Allowed extensions
2 - Addendum

یادآوری ۱- LIPC تضمین نمی‌کند (و نمی‌تواند تضمین کند) که هر رویه معتبری در هر زبانی می‌تواند به وسیله احضار رویه در هر زبان دیگری فراخوانده شود.

در پردازنده زبانی که در طریقه کارساز عمل می‌کند، خدمت LIPC کارساز باید از شکل LIPC ورودی به شکلی که مورد نیاز فراخوانی است (در صورتی به زبان خود کارساز باشد) نامرتب کند. این کار نیاز به نگاشت نوع داده (که در استاندارد LID آمده است) و به احتمال تبدیل نوع داده خواهد داشت.
استانداردِ زبان کارساز، ممکن است خودش امکان برخی از تبدیلات انواع داده‌ای را داشته باشد. به عنوان مثال، در صورتی که پارامتر صوری که فراخوانی با مقدار (در راه‌اندازی) شده است، از نوع عدد حقیقی باشد؛ استاندارد ممکن است پارامتر واقعی از نوع عدد صحیح را نیز بپذیرد، و مقدار عدد صحیح را به شکل ممیز شناور تبدیل کند. بنابر این، در صورتی که خدمت LIPC کارساز، چنین پارامتر واقعی را دریافت کند، می‌توان آن را از نوع عدد صحیح LID به عدد صحیح کارساز تبدیل و مقدارش را ارسال کند. نیازی نیست که استاندارد انقیاد، چیزی در رابطه با این تبدیل نوع داده بگوید؛ زیرا این کار، درست مانند این‌که بخشی از فراخوانی غیر LIPC معمول است، در خود پردازنده کارساز مدیریت می‌شود.
اما حالتی را فرض کنید که پارامتر واقعی ورودی از نوع داده‌ی مقیاس‌دار LID باشد- به‌عنوان مثال به دلیل این‌که زبان کارخواه از ممیز شناور یا دیگر انواع داده‌ای حقیقی تقریبی، پشتیبانی نمی‌کند. در این شرایط استاندارد انقیاد باید مشخص کند که آیا تبدیل عدد مقیاس‌دار LID به عدد حقیقی کارساز مجاز است یا خیر، و اگر مجاز است، نحوه‌ی انجام آن را بیان کند.

یادآوری ۲- بنابر این، تبدیلات نوع داده در یک فراخوانی LIPC به صورت کلی بر دو نوع است – درون خدمت LIPC در طی نامرتب کردن، که در استاندارد انقیاد مشخص می‌شوند. و دیگری درون پردازنده کارساز، در طول راه‌اندازی رویه. ممکن است هر دو نوع تبدیل داده بر روی برخی از پارامترهای واقعی اعمال شوند. کارخواه فراخواننده هیچ تفاوتی را متوجه نمی‌شود.

بنابراین، استاندارد انقیاد برای طریقه کارساز نه تنها باید نگاشت‌های LID برای انواع داده‌ی پارامتر را مشخص کند، بلکه باید به‌طور صریح تبدیلات مجاز از انواع داده‌ای LID ای که معادل مستقیمی در زبان کارساز ندارند را نیز مشخص کند.

یادآوری ۳- استاندارد انقیاد LID ممکن است چنین تبدیلاتی را مشخص، مجاز، یا توصیه کند، اما انقیاد LIPC باید این موضوع را در پرتوی زمینه‌ی ویژه‌ی ارسال پارامتر، بازبینی کند. که ممکن است (همانطور که در خیلی از زبان‌ها این گونه است) منجر به تحمیل محدودیت‌های اضافه‌تری بر انواع داده‌ای مجاز شود.

نکته‌ی ذکر شده، در رابطه با حالت‌های ارسال پارامتر در زبان برای انقیاد طریقه کارخواه است. که البته در مورد سمت کارساز نیز برقرار است. استاندارد انقیاد، باید تضمین کند که ناسازگاری بین این دو وجود ندارد.
انقیاد طریقه کارساز، همچنین باید اجازه‌ی مرتب کردن معکوسِ نتایج بازگشتی (پارامترهای خروجی) حاصل از فراخوانی را هم بدهد.
دو مورد دیگر نیز باید در طریقه کارساز لحاظ شود: پارامترهای از نوع رویه، و «متغیرهای سراسری».

پ-۴ پارامترهاي از نوع رويه (پارامترهاي رويه‌ي)

در صورتی که رویه کارساز، پارامترِ رویه‌ی صوری داشته باشد، به این معنی است که رویه کارساز در طول اجرایش، رویه واقعی داده شده به عنوان پارامتر واقعی را فراخوانی خواهد کرد. ممکن است رویه کارساز (بسته به زبان)، تعداد و انواع داده‌ای مربوط به پارامترهای رویه واقعی داده شده را (به صورت مستقیم یا غیر مستقیم) مشخص کند، یا بتواند با پارامتر رویه‌ی (صوری) به روش عام برخورد کند. از آن‌جایی که انواع داده‌ای از نوع رویه در LID، حاوی اطلاعاتی در مورد تعداد و انواع داده‌ای پارامترهای رویه هستند، دو خدمت LID (سمت کارخواه و سمت کارساز)، برای این‌که بتوانند تصمیم بگیرند که آیا پارامترِ رویه‌ی واقعی داده شده می‌تواند به وسیله کارساز فراخوانی شود یا خیر، می‌توانند اطلاعات لازم را از طریق نگاشت‌های LID پارامترِ رویه‌ی به دست آورند. در صورتی که کارساز تعداد و انواع داده‌ای پارامترهای رویه واقعی را مشخص کند، نامرتب کردن می‌تواند بررسی کند که پارامترِ رویه‌ی ورودی قابل قبول است یا خیر.

در حالت کلی، خودِ فراخوانی یکی از سه نوع زیر خواهد بود:

۱- ممکن است از موضع رویه به سمت کارساز باشد.

۲- ممکن است از موضع رویه به سمت کارخواه باشد.

۳- ممکن است از موضع رویه نه به سمت کارخواه و نه به سمت کارساز باشد، بلکه به سمت کارساز دیگری باشد.

استاندارد انقیاد باید تمامی این موارد را پوشش دهد.

در مورد ۱، کارساز به سادگی رویه را به صورت عادی فراخوانی می‌کند و به کارش ادامه می‌دهد.

در مورد ۲، رویه‌ی که به وسیله پارامترِ رویه‌ی مورد ارجاع قرار گرفته است، سمت موضع کارخواه قرار دارد، و فراخوانی یک فراخوانی معکوس از سمت کارساز (که برای این پارامترِ رویه‌ی در طریقه کارخواه عمل می‌کند) به سمت کارخواه (که برای همین پارامتر رویه‌ی در طریقه کارساز عمل می‌کند) به حساب می‌آید. لذا فقط در صورتی که پردازنده‌ها بتوانند در هر دو حالت عمل کنند، پارامترهای رویه‌ی، می‌توانند پشتیبانی شوند، و قوانین انطباق برای استاندارد انقیاد باید این مورد را منعکس کند.

یادآوری – توجه داشته باشید که، از آن‌جایی که هر دو سمت، منطبق با هر دو طریقه هستند، دو خدمت LIPC می‌توانند بر طریقه‌های ارسال پارامتر برای فراخوانی‌های پارامترهای واقعی از نوع داده‌ای رویه علاوه بر فراخوانی اصلی، توافق کنند.

توجه داشته باشید که حالت ارسال «فراخوانی با مقدار ارسال شده در هنگام تقاضا»، و ارجاع‌ها در بدنه‌ی رویه کارساز به پارامترهای اشاره‌گر، همگی می‌توانند به منزله‌ی فراخوانی‌های معکوسِ «رویه‌های خُرد[1]» تعبیر شوند، اما نیازی نیست این مورد در استاندارد انقیاد صورت پذیرد. این به این دلیل است که پشتیبانی چنین ویژگی‌هایی نیازی به قابلیت فراخوانی رویه به‌طور کامل دوسویه ندارد، و می‌تواند وابسته-به-پیاده‌سازی باشد – (اگرچه به احتمال محدودیت‌ها و شرایط بر آن تأثیر خواهند گذاشت).

در مورد ۳، که در آن رویه ارجاع شده به وسیله پارامترِ رویه‌ی، روی سامانه‌ی سوم قرار دارد، فراخوانی باعث می‌شود که سمت کارساز، نسبت به سامانه‌ی سوم در طریقه کارخواه عمل کند. بنابر این، در این مورد نیز

1 - Mini procedures

سمت کارساز باید بتواند در هر دو طریقه عمل کند، و قوانین انطباق برای استاندارد انقیاد باید این مورد را منعکس کند.

پ-۵ متغیرهاي سراسري

برخی زبان‌ها اجازه‌ی ارجاع به متغیرهای سراسری (هستارهایی که در بدنه‌ی رویه اعلان نشده‌اند، بلکه در محیطی که آن را در بر می‌گیرد از قبیل یک بلاک محاط کننده اعلان شده‌اند) در بدنه‌ی رویه را می‌دهند. در محیط LIPC، اگر این متغیرهای سراسری همیشه سمت کارساز ارائه شده باشند، مشکلی را سبب نمی‌شوند. مشکلات زمانی رخ می‌دهد که فرض می‌شود هستارهای مفقود، در سمت کارخواه ارائه شده‌اند.

استاندارد انقیاد، باید به این سؤال پاسخ دهد. یک راه ساده این است که بگوید رویه کارسازی که به متغیرهای سراسری در سمت کارخواه ارجاع می‌دهند در استاندارد انقیاد مجاز نیست. اما این قید، مواردی که در آن زبان کارخواه و کارساز می‌توانند به فضای ذخیره‌سازی مشترک ارجاع دهند را شامل نمی‌شود و محیط LIPC می‌تواند کماکان از آن‌ها پشتیبانی کند. راه حل دیگر این است که می‌توان چنین متغیرهای سراسری را پارامترهای اضافی مفهومی تصور کرد که (علاوه بر پارامترهای معمولی) در LIPC ارسال می‌شوند. این پارامترهای مفهومی به روش یکسان ارسال پارامترهای معمولی (در قیود قرارداد مجازی) ارسال خواهند شد، اما قرارداد خدمت بین دو خدمت LIPC، آن‌ها را از طریق فضای ذخیره‌سازی مشترک مدیریت خواهد کرد، منوط به این که تأثیرش با زمانی که متغیرهای سراسری با پارامترهای صوری (در مشخصات رویه کارساز) جایگزین می‌شوند، یکسان باشد.

شرایط محیط، هم بین زبان‌ها و هم بین خدمات و محیط‌های LIPC بسیار تغییر خواهد کرد، و رهنمود کلی بر نحوه‌ی حل این سؤال در استاندارد انقیاد نمی‌توان داد. به هر حال، در صورتی که متغیرهای سراسری اعلان نشده، در بدنه‌ی رویه زبان مجاز باشد، استاندارد انقیاد باید آن‌ها را مد نظر قرار دهد.

یادآوری – مدل LIPC ای که در بند ۶ در رابطه با معنای فراخوانی رویه است ممکن است به تصمیم‌گیری در مورد نحوه‌ی مدیریتِ متغیرهای سراسری در استاندارد انقیاد کمک کند.

پيوست ت
(اطلاعاتي)
مروري بر همترازي LIPC IDN – RPC IDL

این پیوست، مفاهیم ضمنی نشانه‌گذاری تعریف واسط LIPC (LIPC IDN) را آن گونه که در این استاندارد ملی تعریف شده‌اند را با مفاهیم ضمنی زبان تعریف واسط RPC (RPC IDN) آن گونه که در بند ۴ استاندارد RPC تعریف شده‌اند (ISO/IEC 11578) مقایسه می‌کند.

مقایسه‌ای در رابطه با شکل‌های نحوی این دو نشانه‌گذاری صورت نگرفته است.

به صورت کلی، LIPC IDN تفاوت‌های معنایی غنی‌تری را نسبت به RPC IDL ارائه می‌کند. هر نوع RPC IDL، شکل معادلی را به صورت انتزاعی در LIPC IDN دارد. یعنی، نوع LIPC مجموعه‌ی مقادیر یکسان با نوع RPC را توصیف می‌کند. اگرچه، RPC IDL می‌تواند موارد نمایشی که خارج از حوزه‌ی LIPC هستند را توصیف کند، و RPC IDL می‌تواند اطلاعات اضافی غیر-نوعی مربوط به مشخصات خدمت RPC خاص را نیز ارائه کند.

اعلان بالاترین سطح، هم در LIPC IDN و هم در RPC IDL، اعلان واسط است. تمام انواع دیگر اعلان‌ها، به عنوان قسمتی از اعلان‌های واسط هستند. مواقعی که فقدان شناسه‌ی واسط در مشخصات منجر به ابهام نمی‌شود، اعلان واسط ممکن است حذف شود، و اعلان 'عادی' به عنوان اعلان بالاترین سطح قرار گیرد.

ت-۱ اعلان‌هاي واسط

مفهوم «واسط» در LIPC و RPC یکسان هستند.

ت-۱-۱ صفات

LIPC IDN اجازه می‌دهد برای تشخیص یکتای واسط از OSI object-identifier استفاده شود. RPC IDL یک صفت uuid(X) دارد که عمل یکسانی را انجام می‌دهد (با وجود این‌که X یک OSI object-identifier نیست).

RPC IDL سه صفت (نسخه[1]، نقطه‌ی پایانی[2] و محلی[3]) دارد که جهت ارائه‌ی یک خدمت RPC در RPC IDL استفاده شده‌اند. این سه ویژگی معادلی در LIPC ندارند.

RPC IDL صفتی به نام pointer_default دارد که اجازه‌ی حذف صفات اشاره‌گر معینی در بدنه‌ی واسط را می‌دهد. این امر فقط جهت سهولت در نشانه‌گذاری است.

ت-۱-۲ بند واردات[4]

هم LIPC IDN و هم RPC IDL اجازه می‌دهند اسامی از واسط دیگری (که از قبل موجود است) به اعلان واسط جاری وارد شوند. بدین وسیله اسامی می‌توانند در بدنه‌ی واسط جاری استفاده شوند.

LIPC IDN می‌تواند اسامی وارد شده از واسط از پیش موجود را (به وسیله فهرست کردن صریح اسامی مطلوب) محدود کند. اما RPC IDL نمی‌تواند.

---

1 - Version
2 - Endpoint
3 - Local
4 - Imports clause

LIPC IDN اجازه می‌دهد اسامی وارد شده با اسم واسط منبع مشخص شوند. اما RPC IDL این امکان را ندارد.

ت-۲ اعلان‌هاي ديگر

هم LIPC IDN و هم RPC IDL، اجازه‌ی اعلانِ انواع دارای نام[1]، مقادیر دارای نام، و رویه‌های (عملیات) دارای نام را به واسط‌ها می‌دهند. به علاوه، در LIPC IDN، شرایط پایان‌دهی باید دارای نام شوند.

ت-۲-۱ اعلان‌هاي نوع

هم LIPC IDN و هم RPC IDL می‌توانند نامی را به هر نوعی که در نشانه‌گذاری قابل تعریف است، همبسته سازند.

LIPC IDN بین انتساب یک مترادف به یک نوع موجود، و تعریف یک نام برای یک نوع جدید تفاوت قائل می‌شود. RPC IDL این تفاوت را در حالت کلی قائل نمی‌شود، اما تفاوت را برای انواع ساخت‌یافته و انواع اجتماع (union) مجاز می‌داند. (مفهوم «tagged_declarator» در RPC IDL را ببینید.)

LIPC IDN اجازه‌ی تعریف مولدهای نوع (انواع پارامتری شده) را (هم به عنوان مترادف‌ها و هم به عنوان انواع جدید) می‌دهد. اما RPC IDL این اجازه را نمی‌دهد.

در ادامه، انواع داده‌ی قابل تعریف در LIPC IDN و RPC IDL آورده شده است.

ت-۲-۲ اعلان‌هاي مقدار

هم LIPC IDN و هم RPC IDL می‌توانند نامی را به یک مقدار همبسته کنند.

LIPC IDN اجازه می‌دهد اسامی به هر نوعی از مقدار داده شود، و امکان نشانه‌گذاری لفظی برای تمام انواعِ مقادیر را ارائه می‌کند.

RPC IDL اجازه می‌دهد اسامی به مقادیرِ مجموعه‌ی محدودی از انواع اختصاص یابد. انواع مجاز: عدد صحیح، بولی، نویسه، اشاره‌گر-به-نویسه (رشته)، و اشاره‌گر-به-تهی هستند.

RPC IDL اجازه‌ی محاسبه‌ی مقادیر پیچیده را نیز می‌دهد. اما LIPC IDN این امکان را ندارد.

ت-۲-۳ اعلان‌هاي رويه

هم LIPC IDN و هم RPC IDL می‌توانند رویه‌های (عملیات) دارای نام را اعلان کنند.

در هر دو IDN، رویه‌ها می‌توانند صفر یا بیشتر از صفر پارامتر و یک نوع بازگشتی اختیاری داشته باشند. هر پارامتر یک «جهت» و یک نوع دارد. در RPC IDL جهات مجاز: ورودی و خروجی هستند. در LIPC IDN جهات مجاز: ورودی، خروجی، و ورودی-خروجی هستند. جهت ورودی-خروجی معادل مشخصات دو پارامتر (یکی ورودی و یکی خروجی) که هر دو نوع یکسانی دارند، است.

RPC IDL پارامترهای خروجی را به آرایه‌ها یا اشاره‌گرها محدود می‌کند. زیرا RPC IDL آن‌ها را انواع مناسب نسبت‌دهی می‌داند. کاری که با مقدار بازگشتی (خروجی) انجام می‌شود، برای LIPC IDN اهمیتی ندارد. لذا LIPC IDN نوع یک پارامتر خروجی را محدود نمی‌کند. در عمل، تناظرهای زیر بین انواع پارامتر برقرار است:

---

1 - Named Types

RPC parameter type	LIPC parameter type
T*	T
T[…]	array(integer range(…)) of (T)

این تناظر، یک به یک نیست، انتخاب بهترین تطابق نیازمند دانشی از معنی‌شناسی کاربرد است.
اعلان رویه در LIPC IDN، باید تمامی شرایط پایان‌دهی (استثناءها، خطاها) که ممکن است در قسمتی از معنی‌شناسی رویه رخ دهند را فهرست کند. اما RPC IDL این‌گونه نیست. به اعلان‌های پایان‌دهی در بخش ت-۲-۴ مراجعه شود.
RPC IDL، سه صفت مختص RPC قابل اعمال به رویه‌ها دارد: همانی (idempotent)، همه‌پخشی (broadcast)، شاید (maybe).

ت-۲-۴ اعلان‌هاي پايان‌دهي

در LIPC، یک رویه می‌تواند به صورت عادی یا به شکل یکی از پایان‌دهی‌های دارای نام، بازگردد. در بازگشت عادی، مقادیر پارامترهای خروجی و ورودی-خروجی جدید و هر مقدار بازگشتی صریح، به وسیله رویه ارائه می‌شوند. در پایان‌دهی دارای نام، مجموعه‌ی متفاوتی از مقادیر تهیه می‌شوند: آن‌هایی که در اعلان پایان‌دهی، اعلان شده‌اند.
یک اعلان پایان‌دهی LIPC IDN، حاوی نامی برای پایان‌دهی است، و یک فهرست از صفر یا بیشتر از صفر نوع، برای مقادیر بازگشتی است.
RPC IDL ابزاری جهت اعلان شرایط بازگشت غیرعادی ارائه نمی‌کند.

ت-۳ انواع داده‌ي اوليه

ت-۳-۱ بولي (boolean)

نوع بولی LIPC IDN و نوع بولی RPC IDL یکسان هستند.

ت-۳-۲ حالت (state)

مولد نوع حالت LIPC IDN، با مولد نوع شمارشی در RPC IDL یکسان است، با این تفاوت که مقادیر نوع داده‌ی حالت، بدون ترتیب هستند.
در ترجمه‌ی نوع داده‌ی شمارشی RPC به LIPC IDN، باید دانشی از معنی‌شناسی کاربرد جهت انتخاب بهترین تطابق (حالت در مقایسه با شمارشی) داشت.

ت-۳-۳ شمارشي (enumerated)

مولد نوع شمارشی (enumerated) در LIPC IDN و مولد نوع شمارشی (enum) در RPC IDL یکسان هستند.
در ترجمه‌ی نوع داده‌ی شمارشی RPC به LIPC IDN، باید دانشی از معنی‌شناسی کاربرد جهت انتخاب بهترین تطابق (حالت در برابر شمارشی) داشت.

ت-۳-۴ نويسه (character)

نوع داده‌ی نویسه LIPC IDN (R)، شامل تمام نویسه‌های مجموعه نویسه‌ی استاندارد R می‌شود.

چهار نوع داده‌ی نویسه RPC IDL وجود دارد: char، ISO_LATIN1، ISO_UCS و SO_MULTI_LINGUAL.

نوع داده‌ی نویسه RPC IDL به ظاهر در پیاده‌سازی تعریف می‌شود، اما حداقل شامل مجموعه نویسه ISO 646 می‌شود. بنابراین فقط درصورتی که نویسه قابل جابه‌جایی باشد، داریم:

RPC type	LIPC equivalent
char	character (iso standard 646)
ISO_LATIN_1	character (iso(1) standard(O) 8859 part(1))
ISO_UCS	character (iso(1) standard(O) 10646)
ISO_MULTLLINGUAL	character (iso(1) standard(O) 10646 multi-lingual-plane)

ت-۳-۵ عدد وصفي (ordinal)

نوع عدد وصفی LIPC IDN، شبیه به محدوده‌ی (..0) عدد صحیح است. نزدیک‌ترین معادل در RPC IDL، hyper بدون علامت است.

ت-۳-۶ زمان (time)

RPC IDL، معادلی برای مولد نوع زمانِ LIPC IDN ندارد.

یک کدبندی ممکن از نوع time(unit,radix,factor) در LIPC IDN به نوع RPC IDL: hyper ای است که مقدار hyper، h، زمانی که به عنوان مقدار نوع time(unit,radix,factor) دیده شود، تفسیر زیر را دارد:

$$h*(radix^{-factor})*unit$$

ت-۳-۷ عدد صحیح (integer)

LIPC IDN یک نوع عدد صحیح بیکران دارد، در حالی که RPC IDL هشت نوع عدد صحیح دارد که در ادامه آورده شده است:

RPC type	LIPC equivalent
hyper	integer range (- 2^{63} .. 2^{63}_1)
long	integer range (_2^{31} .. 2^{31}_1)
short	integer range (- 2^{15} .. 2^{15}_1)
small	integer range (_2^{7} .. 2^{7}-1)
unsigned hyper	integer range (0 .. 2^{64}-1)
unsigned long	integer range (0 .. 2^{32} -1)
unsigned short	integer range (0 .. 2^{16}_1)
unsigned small	integer range (0 .. 2^{8} -1)

در ترجمه‌ی نوع داده‌ی عدد صحیح LIPC به RPC IDL، باید دانشی از معنی‌شناسی کاربرد جهت انتخاب بهترین تطابق داشت. ترجمه‌ی بدون نقص امکان‌پذیر نیست.

ت-۳-۸ عدد گویا (rational)

RPC IDL معادلی برای نوع داده‌ی گویا در LIPC IDN ندارد.

کدبندی زیر یک کدبندی ممکن برای نوع عدد گویای LIPC IDN به نوعی در RPC IDL است:

struct { hyper numerator, denominator; }

که در آن مقسوم علیه (denominator) بزرگتر از صفر است و مقسوم (numerator) و مقسوم علیه نسبت به هم اول هستند.

ت-۳-۹ داده‌ي مقياس‌دار (scaled)

RPC IDL معادلی برای مولد نوع مقیاس‌دار در LIPC IDN ندارد.

یک کدبندی ممکن از نوع scaled(radix, factor) در LIPC IDN به یک نوع در RPC IDL، hyperای است که مقدار hyper، h، زمانی که به عنوان مقداری از نوع scaled(radix, factor) دیده شود، تفسیر زیر را دارد:

$$h * (radix^{-factor})$$

ت-۳-۱۰ عدد حقيقي (real)

LIPC IDN، با دانه‌بندی[1] تقریب داده شده‌ی $(radix^{-factor})$، مولد نوع عدد حقیقی را ارائه می‌کند.

RPC IDL دو تقریب برای اعداد حقیقی ارائه می‌کند: float و double. دقت و بُرد این دو نوع نامشخص است، اگرچه ممکن است شبیه به انواع single و double در IEC 559 باشند. اگر این‌گونه باشد، معادل LIPC این انواع به شکل زیر خواهد بود:

RPC type	LIPC analog
float	real (2,24)
double	real (2,53)

ت-۳-۱۱ اعداد مختلط (complex)

RPC IDL معادلی برای مولد نوع مختلط در LIPC IDN ندارد.

یک کدبندی ممکن از نوع complex (radix, factor) در LIPC IDN به یک نوع در RPC IDL یکی از موارد زیر است:

struct { float real_part, imaginary -part; }

struct { double real_part, imaginary-part; }

بر اساس این‌که کدام نوع float یا double تقریب بهتری برای real(radix,factor) است، یکی از آن‌ها انتخاب می‌شود.

ت-۳-۱۲ تهي (void)

LIPC IDN و RPC IDL نوع تهی یکسانی را ارائه می‌کنند. اما استفاده از تهی در RPC IDL به (۱) مقصد یک اشاره‌گر، یا (۲) نوع بازگشتی یک رویه، محدود شده است.

اشاره‌گر به تهی به نظر کاربرد کمی دارد. اما در RPC IDL فرض ضمنی (که به احتمال ایده‌ی آن از زبان C گرفته شده است) وجود دارد مبنی بر این‌که هر *T ای می‌تواند به و از یک *void بدون از دست دادن اطلاعات منتقل شود. بنابر این، *void را می‌توان جهت ارسال داده‌ی اشاره‌گر در یک واسط استفاده کرد. متناظر LIPC IDN این مفهوم، محرمانه است.

ت-۴ توصيف‌گرهاي نوع

RPC IDL امکانات متناظری برای شش توصیف‌گر نوع تعریف شده در LIPC IDN: Range، Selecting، Excluding، Extended، Size و Subtype، ندارد.

ت-۵ داده هاي توليد شده

1 - Granularity

ت-۵-۱ انتخاب (choice)

مولد نوع انتخاب در LIPC IDN زوجِ مقدار برچسب و مقدارِ یکی از چندین نوعِ عنصر جایگزین را تولید می‌کند. مقدار برچسب مشخص می‌کند کدام یک از انواع عناصر جایگزین استفاده می‌شوند.

مولد نوع اجتماع (union) در RPC IDL، معنی‌شناسی یکسانی با مولد نوع انتخاب در LIPC IDN دارد. با این تفاوت که (در RPC IDL) نوع برچسب محدود به نوع عدد صحیح، بولی، نویسه، یا نوع شمارشی است. در LIPC IDN نوع برچسب می‌تواند هر نوع داده‌ی دقیقی باشد. نوع داده‌ی دقیق، نوع داده‌ای است که عدد حقیقی و مختلط نباشد و از عدد حقیقی یا مختلط نیز تولید نشده باشد.

هم LIPC IDN و هم RPC IDL اجازه می‌دهند بخش برچسب یک اجتماع یا انتخاب از مقدار حذف شود. این کار فقط زمانی انجام می‌شود که انتخاب یا اجتماع، جزئی از نوع داده‌ی بزرگتری (یا فهرست پارامتر) باشند که یکی از فیلدهایش مقدار برچسب است.

در RPC IDL استفاده از آرایه‌ی منطبق[1] یا منطبق متغیر[2] به عنوان یک گزینه‌ی union مجاز نیست. LIPC IDN چنین محدودیت‌هایی ندارد.

توجه داشته باشید که نحوِ RPC IDL انواع union دارای نام جدید را پشتیبانی می‌کند.

ت-۵-۲ اشاره‌گر

مولد نوع اشاره‌گر به (T) در LIPC IDN و مولد نوع *T در RPC IDL، به‌طور اساسی معنی‌شناسی یکسانی دارند. اگرچه، RPC IDL دو صفت برای تغییر معنای اشاره‌گرها دارد و به طرز قابل توجهی کاربرد اشاره‌گرها را محدود می‌کند.

RPC type	LIPC equivalent
[ptr] T*	pointer to (T)
[ref] T*	pointer to (T) excluding (null)

صفت [ref] در RPC IDL اطلاعاتی را نیز برای یک خدمت RPC فراهم می‌کنند، به‌عنوان مثال گزاره‌ای دارد که نشان می‌دهد هیچ مستعاری برای داده‌ای که به وسیله اشاره‌گر در طول فراخوانی RPC مورد ارجاع قرار گرفته است، وجود ندارد.

RPC IDL اشاره‌گرهای معینی را معادل آرایه‌ها یا جانشین‌های آرایه‌ها در نظر می‌گیرد. لذا در ترجمه‌ی انواع اشاره‌گر RPC IDL به LIPC IDN، دانشی از معنی‌شناسی کاربرد و جزئیات قوانین RPC IDL، جهت انتخاب بهترین تطابق (اشاره‌گر و آرایه) نیاز خواهد بود.

اعلان نوع (ت-۲-۱) و بندهای رویه را برای توضیحات بیشتر در رابطه با اشاره‌گرها ببینید.

ت-۵-۳ رویه

RPC IDL انواع از نوع رویه را پشتیبانی نمی‌کند. رویه‌ها در LIPC تحت عنوان عملیات در RPC تعریف شده‌اند (به بند ت-۲-۳ مراجعه شود).

ت-۶ انواع داده‌ي انبوهشي

1 - Conformant
2 - Conformant-varying

ت-۶-۱ رکورد (record)

مولد نوع رکورد در LIPC IDN و مولد نوع ساختار (struct) در RPC IDL یکسان هستند. توجه داشته باشید که نحوِ RPC IDL اجازه‌ی انواع ساختار دارای‌نام جدید را می‌دهد.

ت-۶-۲ مجموعه (set)

در RPC IDL مولد نوع مجموعه وجود ندارد. مفهوم مجموعه در LIPC IDN می‌تواند با یک آرایه در RPC IDL مدل شود. برای مثال:

LIPC type	RPC analog
set of (T)	struct { long size; T element [size]; }

بنابر این، در ترجمه‌ی نوع آرایه RPC IDL به LIPC IDN، دانشی از معنی‌شناسی کاربرد جهت تشخیص این‌که آیا چیزی که به‌طور واقعی نیاز است یک مجموعه است یا خیر، ضروری است.

ت-۶-۳ کیسه (bag)

در RPC IDL مولد نوع bag وجود ندارد. مفهوم bag در LIPC IDN را می‌توان با یک آرایه در RPC IDL مدل کرد. برای مثال:

LIPC type	RPC analog
bag of (T)	struct { long size; T element [size]; }

بنا بر این، در ترجمه‌ی نوع آرایه RPC IDL به LIPC IDN، دانشی از معنی‌شناسی کاربرد جهت تشخیص این‌که آیا چیزی که به‌طور واقعی نیاز است یک bag است یا خیر، ضروری است.

ت-۶-۴ دنباله (sequence)

در RPC IDL مولد نوع دنباله وجود ندارد. مفهوم دنباله در LIPC IDN را می‌توان با یک آرایه در RPC IDL مدل کرد. برای مثال:

LIPC type	RPC analog
sequence of (T)	struct { long size; T element [size]; }

بنابر این، در ترجمه‌ی نوع آرایه RPC IDL به LIPC IDN، دانشی از معنی‌شناسی کاربرد جهت تشخیص این‌که آیا چیزی که به‌طور واقعی نیاز است یک دنباله است یا خیر، ضروری است.

ت-۶-۵ آرایه (array)

در LIPC IDN، مولد نوع array (I) of (T)، مجموعه‌ی نمایه گذاری شده‌ای از مقادیر را تعریف می‌کند. مقدار آرایه در LIPC مقدار عنصر T را به مقدار شاخص I همبسته می‌کند. بنابر این، تمام مقادیر نوع آرایه در LIPC «طول» یکسانی دارند که به وسیله اندازه‌ی شاخص نوع I تعیین می‌شود. نوع شاخص می‌تواند هر نوع محدودی باشد.

مولد نوع آرایه در RPC IDL نیز مجموعه‌ی نمایه گذاری شده‌ای از مقادیر را تعریف می‌کند. اما، فقط محدوده‌ای از اعداد صحیح را برای نوع شاخص می‌پذیرد.

RPC type	LIPC equivalent
T [a..b]	array (integer range (a .. b)) of (T)

هم LIPC IDN و هم RPC IDL اجازه می‌دهند آرایه‌ها چند بُعدی باشند (انواع نمایه).

هم در LIPC IDN و هم در RPC IDL، حدود آرایه می‌تواند به وسیله مقادیر وابسته معین شوند. بند مقادیر وابسته (ت-۹) که در ادامه آمده است را ببینید.

RPC IDL بین حدود ذخیره‌سازی آرایه و حدود داده‌ی-معتبر همان آرایه تفاوت قائل می‌شود. مشخصات LIPC IDN مستقل از این‌چنین اطلاعات مربوط به نمایش است.

یک آرایه‌ی سازگار در RPC IDL، آرایه‌ای است که از مقادیر وابسته برای مشخص کردن حدود ذخیره‌سازی آرایه استفاده می‌کند. یک آرایه‌ی متغیر در RPC IDL، آرایه‌ای است که از مقادیر وابسته برای مشخص کردن حدود داده‌ی معتبر آرایه استفاده می‌کند.

از آن‌جایی که RPC IDL بین دنباله‌ها، کیسه‌ها، جداول و آرایه‌ها تفاوتی قائل نمی‌شود، ممکن است برخی از کاربردهای آرایه در RPC بیشتر به عنوان دنباله‌ها دیده شوند. برای کسب اطلاعات بیشتر به بند دنباله‌ها (ت-۶-۴)، کیسه‌ها (ت-۶-۳)، و جداول (ت-۶-۶) مراجعه شود.

ت-۶-۶ جدول

RPC IDL مولد نوع جدول ندارد. مفهوم جدول در LIPC IDN را می‌توان با آرایه‌ای از ساختار در RPC IDL مدل کرد. برای مثال:

LIPC type	RPC analog
table (FL)	struct { long size; struct {FL'} row [size]; }

که در آن 'FL معادل RPC فهرست فیلد FL در LIPC است.

بنابر این، در ترجمه‌ی نوع آرایه RPC IDL به LIPC IDN، دانشی از معنی‌شناسی کاربرد جهت تشخیص این‌که آیا چیزی که به‌طور واقعی نیاز است یک جدول است یا خیر، ضروری است.

ت-۷ مولدها و انواع داده‌ي مشتق شده

ت-۷-۱ عدد طبيعي (natural number)

معادل RPC IDL برای نوع naturalnumber، hyper بدون علامت است.

ت-۷-۲ همنهشتي (پيمانه) (modulo)

معادل RPC IDL برای نوع modulo(n)، نوع عدد صحیح بدون علامتی است که به اندازه‌ی کافی برای نمایش مقادیر 0 تا n-1 بزرگ است.

ت-۷-۳ بيت (bit)

معادل RPC IDL برای نوع بیت، بولی است.

ت-۷-۴ رشته‌بيتي (bitstring)

معادل RPC IDL برای نوع رشته‌بیتی به شکل زیر است:

struct { long size; T element [size]; }

در صورتی که رشته‌بیتی اندازه‌ی ثابتی داشته باشد، بهترین معادل RPC IDL یک آرایه‌ی بولی خواهد بود.

ت-۷-۵ رشته‌نويسه‌اي (characterstring)

نوع داده‌ی characterstring(R) در LIPC IDN حاوی رشته‌هایی (دنباله‌هایی) از نویسه‌هایی است که از نوع character(R) هستند. لذا، رشته‌ها روی هر مجموعه نویسه‌ای می‌توانند تعریف شوند.

RPC IDL می‌تواند رشته‌هایی از نویسه‌ها، بایت‌ها، یا ساختارهایی که فقط شامل بایت‌ها هستند را تعریف کند. اشکال آخر جهت مدیریت فهرست نمایش نویسه‌هایی به غیر از ISO 646 (در سطح نمایش به جای سطح منطقی) به کار برده می‌شوند.

RPC type	LIPC equivalent
[string] char*	characterstring(ISO-646)

در ترجمه‌ی انواع رشته از LIPC IDN به RPC IDL مفهوم منطقی فهرست نمایش نویسه باید با یک نمایش خاص که مربوط به بایت‌ها است، جایگزین شود. این امر نیازمند انتخاب یک نمایش خاص است.

در ترجمه‌ی معکوس، دانشی از معنی‌شناسی کاربرد جهت تعیین فهرست نمایش نویسه مورد نظر، نیاز خواهد بود.

ت-۷-۶ فاصله‌ي زماني (timeinterval)

RPC IDL، معادلی برای مولد نوع فاصله‌ی زمانی در LIPC IDN ندارد. یک کدبندی محتمل از نوع timeinterval(unit,radix,factor) به یک نوع RPC IDL، می‌تواند hyper ای باشد که در آن مقدار hyper از h، زمانی که به عنوان یک مقدار از نوع timeinterval(unit,radix,factor) باشد، تفسیر زیر را دارد:

$$h*(radix^{-factor})*unit$$

ت-۷-۷ هشت‌تايي (octet)

نوع هشت‌تایی در LIPC IDN با نوع small بدون علامت در RPC IDL یکسان است.

ت-۷-۸ رشته‌ي هشت‌تايي (octet string)

معادل نوع رشته‌ی هشت‌تایی در RPC IDL عبارت زیر خواهد بود:

struct { long size; unsigned small element [size]; }

در صورتی که رشته‌ی هشت‌تایی اندازه‌ی ثابتی داشته باشد، بهترین معادل RPC IDL آرایه‌ای از small بدون علامت خواهد بود.

ت-۷-۹ خصوصي (private)

نوع داده‌ی private (n) در LIPC IDN حاوی داده‌های مبهم[1] با طول n بیت است.

نوع داده‌ی بایت در RPC IDL داده‌های مبهم دارد که به احتمال طولشان هشت بیت است. بنابر این:

1 - Opaque

RPC type	LIPC equivalent
Byte	private(8)

برخی از کاربردهای بایت ممکن است بیشتر متناظر با نوع هشت‌تایی در LIPC باشد (که مبهم نیست).

ت-۸ انواع داده‌اي ديگر در RPC

انواع داده‌ای handle_t، pipe T، و context_handle در RPC مختص خدمت RPC هستند، و مبهم هستند. این انواع معادل مستقیمی در LIPC ندارند.

انواع مختص خدمت مبهم به صورت انتزاعی به عنوان انواع خصوصی جدید به بهترین شکل مدیریت می‌شوند.

نوع داده‌ی error_status_t در RPC مختص خدمت RPC است، اما مبهم نیست.

RPC type	LIPC equivalent
error_status_t	integer range (0 .. 2^{32}_1)

ت-۹ «مقادیر وابسته»

یک مقدار «وابسته»، شناسه‌ای است که در مشخصات نوع فیلد (از رکورد) یا یک پارامتر (از یک رویه) استفاده می‌شود و مقدار ثابتی ندارد. این مقدار نام یک فیلد در رکورد در بر گیرنده، یا نام یک پارامتر در فهرست پارامتر رویه است.

ت-۱۰ مراجع متقابل

مبحث بالا در ارتباط با انواع داده‌ای LIPC سازماندهی شده است. جدول زیر، نوع داده‌ای RPC و بندی که در آن بحث شده است را نشان می‌دهد.

جدول ت-۱- نمایه ارجاعات به انواع داده‌ای

نوع داده‌ی RPC	بند مربوطه
boolean	Boolean (ت-۳-۱)
byte	Octet (ت-۷-۷)، Private (ت-۷-۹)
char	Character (ت-۳-۴)
ISO_LATIN_l	Character (ت-۳-۴)
ISO_UCS	Character (ت-۳-۴)
ISO_MULTLLINGUAL	Character (ت-۳-۴)
small	Integer (ت-۳-۷)
short	Integer (ت-۳-۷)
long	Integer (ت-۳-۷)
hyper	Integer (ت-۳-۷)
unsigned small	Integer (ت-۳-۷)
unsigned short	Integer (ت-۳-۷)
unsigned long	Integer (ت-۳-۷)
unsigned hyper	Integer (ت-۳-۷)
float	Real (ت-۳-۱۰)
double	Real (ت-۳-۱۰)
pointer	Pointer (ت-۵-۲)
array	Array (ت-۶-۵)، Sequence (ت-۶-۴)، Set (ت-۶-۲)
	Bag (ت-۶-۶)، Table (ت-۶-۶)
string	Characterstring (ت-۷-۵)
enum	Enumerated (ت-۳-۳)، State (ت-۳-۲)
struct	Record (ت-۶-۱)
union	Choice (ت-۵-۱)
procedure	Procedure (ت-۵-۳)
void	Void (ت-۳-۱۲)
context_handle	Other RPC Datatypes (ت-۸)
handle	Other RPC Datatypes (ت-۸)
pipe	Other RPC Datatypes (ت-۸)
error status	Termination Declarations (ت-۲-۲۴)، Other RPC Datatypes (ت-۸)