

Student Record Exchange   
SIF Zone Services

SIF Service Proposal

Student Record Exchange Task Force

Revision 1.0

April 27, 2009

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| DRAFT 0.1 | 11/12/2008 | Eric Petersen Edustructures | Initial draft. |
| DRAFT 0.2 | 11/18/2008 | Eric Petersen Edustructures | Update to initial draft. |
| DRAFT 0.3 | 11/19/2008 | Eric Petersen Edustructures | Added Section 1.2 *Student and Agency Identifiers* and 1.3 *Security*; added *Appendix*; changed font (step numbers not visible on Windows machines) |
| DRAFT 0.4 | 2/19/2009 | Eric Petersen Edustructures | Added StudentRecordExchangeDataList element to contain repeatable StudentRecordExchangeData elements; added SIF\_Properties to all service methods |
| DRAFT 0.5 | 3/11/2009 | Eric Petersen Edustructures | Reversed one change made in 0.4: removed the StudentRecordExchangeDataList container element |
| 1.0 | 4/27/2009 | Eric Petersen Edustructures | Removed all [TBD] sections for SIF 2.4 Specification; changed all occurrences of SIF\_Properties element to ExtendedParameters |

# Table of Contents

Revision History 2

Table of Contents 2

1 Overview 5

1.1 Terminology 5

1.2 Student and Agency Identifiers 6

1.3 Services Overview 6

1.4 Intermediary Agents 7

1.5 Transaction Services 8

1.5.1 Summary of Service Methods 9

1.5.2 Requesting Student Records Directly from a Trading Partner 10

1.5.3 Requesting Student Records via a Brokerage 11

1.5.4 End-to-End Student Record Exchange from a Brokerage 12

1.6 Consumer Services 14

1.6.1 Rules for Calling Consumer Services 15

1.6.2 Consuming Student Records into Target Applications 16

1.7 Web Services 18

2 Use Cases 20

2.1 Use Case SRE01 – Brokered LEA-to-LEA Student Records Transfer 20

2.2 Use Case SRE02 – Brokered LEA-to-Postsecondary Transcript Exchange 20

3 Service Definition 21

3.3 The StudentRecordExchangeData Composite Object 21

3.3.1 SIF\_MaxBufferSize and StudentRecordExchangeData 22

3.3.2 ExtendedParameters 22

3.4 SREBrokerage Service 23

3.4.1 Service Methods 23

3.4.2 Service Events 23

3.4.3 Detailed Message Definition 23

3.4.4 GetStudentRecordExchange Message 23

3.4.5 GetStudentRecordExchangeResponse Message 23

3.5 SREPublisher Service Definition 24

3.5.1 Service Methods 24

3.5.2 Service Events 24

3.5.3 Detailed Message Definition 24

3.5.4 PublishStudentRecordExchange Message 24

3.5.5 PublishStudentRecordExchangeResponse Message 25

3.6 SREConsumer Service Definition 25

3.6.1 Service Methods 25

3.6.2 Service Events 25

3.6.3 Detailed Message Definition 25

3.6.4 ConsumeStudentRecordExchange Message 26

3.6.5 ConsumeStudentRecordExchangeResponse Message 26

3.7 SRERecordTypeConsumer Service Pattern 27

3.7.1 Service Methods 27

3.7.2 Service Events 27

3.7.3 Detailed Message Definition 27

3.7.4 Consume*RecordType* Message 27

3.7.5 Consume*RecordType*Response Message 28

# 1 Overview

Student Record Exchange (SRE) Services are comprised of six SIF Zone Services that enable trading partners to publish, exchange, and consume electronic transcripts and student records over the SIF infrastructure. The services support both direct and brokered models of student record exchange.

The primary goals of the Student Record Exchange Services are:

* To enable Student Record Exchange choreography between trading partners and brokerages, accommodating all of the use cases identified by the Student Record Exchange Task Force. Using the set of core services defined here, an agent should be able to participate in any of these use cases without knowledge of how a broader student record exchange system is constructed or deployed.
* To enable Student Record Exchange choreography in a way that can be accomplished with or without an intermediary brokerage. Brokerages are used to conduct electronic transcript and student records transfer between trading partners that do not know of one another; for example, to implement LEA-to-LEA student transfer or LEA-to-Postsecondary transcript exchange.
* Provide an unambiguous method of implementing the “last mile” of student record exchange – specifically LEA-to-LEA student records transfer – whereby applications that are authoritative for student records must also be able to consume those records.

## Terminology

The term *trading partner* in this document encompasses all types of organizations that might wish to participate in student record exchanges, including LEAs, SEAs, colleges and universities, non-educational organizations, and so on. Student Record Exchanges are carried out between trading partners. It’s important to keep in mind that a trading partner might represent more than one agency; for example, a data warehouse maintained by a SEA might contain transcript data for many students from many LEAs.

The term *brokerage* means an intermediary system that acts as a broker between trading partners for the purpose of exchanging transcripts or student records. A brokerage may itself be considered a trading partner.

In this document, the term *agency* usually means an LEA. The *sending agency* is the LEA that is furnishing student records, and the *receiving agency* is the LEA that is receiving those records.

## Student and Agency Identifiers

All service messages and objects defined in this document identify students and agencies by StateProvinceId.

A StateProvinceId is a unique identifier that is assigned outside the scope of SIF, typically by a state department of education. For students, the StateProvinceId can be obtained from the StudentPersonal/StateProvinceId element, as well as from the StudentRecordExchange/StateProvinceId element. For agencies that are represented by a LEAInfo object, it can be obtained from the LEAInfo/StateProvinceId element.

It is the responsibility of agents implementing these services to resolve StateProvinceId to a local identifier when needed. Further, because local identifiers are not unique outside of the organization that assigns them, it is not feasible to use local identifiers or SIF RefIds in student record exchange transactions, which span multiple trading partners. It is therefore assumed that any trading partner participating in a student record exchange program has obtained unique StateProvinceId identifiers for students involved in SRE transactions.

## Services Overview

Student Record Exchange Services are divided into two categories: Transaction Services and Consumer Services.

*Transaction Services* allow trading partners to publish and receive StudentRecordExchange objects at the request of another trading partner or brokerage. They offer a transactional, service-oriented alternative to the data-oriented SIF\_Request/SIF\_Response model of exchanging data. Transaction Services are intended to be implemented by intermediary agents that collaborate to form a comprehensive student record exchange system; providers of student data such as SIS systems do not typically implement these services. Transaction Services include:

* SREPublisher
* SREConsumer
* SREBrokerage

*Consumer Services* offer finer-grained control over importing student data into target systems such as SIS and special education packages. Whereas the Transaction Services deal in StudentRecordExchange objects, the optional Consumer Services each address a specific record type contained in that object. The services are organized by record type so that different consumer systems can import select parts of a StudentRecordExchange object independent of other consumers in a zone. Consumer Services are intended to be implemented by local providers of student data such as SIS systems. They include:

* SREStudentDemographicRecordConsumer
* SREStudentAcademicRecordConsumer
* SREStudentSpecialEducationRecordConsumer

## Intermediary Agents

In much the same way as SIF Vertical Reporting solutions rely on intermediary agents that collaborate to implement the overall vertical reporting choreography, a comprehensive student record exchange solution will also make use of intermediary agents in most cases. For example, a commercial state-wide solution might employ several components that collaborate to implement electronic transcript and student records transfer:

* A “brokerage” at the state level to manage communication between trading partners that do not directly know of one another
* A “publisher agent” at the LEA level to publish student record exchanges at the request of the brokerage, or directly from other consumer agents. This intermediary might work with local SIF Zones to query and gather data that is needed to fulfil a StudentRecordExchange request.
* A “consumer agent” at the LEA level to consume student record exchanges received from the brokerage, or directly from other publisher agents. This intermediary might work with local SIF Zones to consume data into target systems such as student information systems.

While the Student Record Exchange Services defined here do not require intermediary agents – any two agents that implement the SREPublisher and SREConsumer services can directly exchange student records – they are designed to not only accommodate but to promote their use.

There are numerous benefits to using intermediary agents. First and foremost, intermediary agents remove the burden of specializing in Student Record Exchange choreography from the broader audience of SIF Agents. By implementing only the SREStudentDemographicRecordConsumer service, for example, a student information system SIF Agent can participate in basic LEA-to-LEA student records transfer without the vendor having to know anything about how the overall system is implemented (which may vary widely from state to state). This is particularly important in realizing widespread adoption of student record exchange capabilities in the marketplace.

Another important benefit to using intermediary agents is that they make it possible to build a comprehensive student record exchange solution with a consistent feature set, user experience, and level of service regardless of the mix of SIF Agents that are used in the overall solution.

Figure 1 illustrates the use of intermediary agents in an overall student record exchange solution:

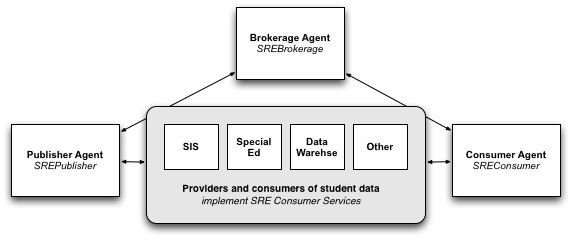


Figure . Intermediary Agents

## Transaction Services

The SRE Transaction Services are comprised of SREPublisher, SREConsumer, and SREBrokerage.

The *SREPublisher* and *SREConsumer* services allow for direct exchange of student records between trading partners or the indirect exchange between trading partners and a brokerage. The SREPublisher service represents a trading partner that can furnish student records. Its PublishStudentRecordExchange method is called to request the records for a given student and sending agency. The SREConsumer service represents a trading partner that can consume student records. Its ConsumeStudentRecordExchange method is called to “push” student records to the trading partner. It is worth repeating here that trading partners do not always represent LEAs, nor do they always represent the ultimate source or destination of data. A state department of education, for example, might act as a trading partner to furnish student records from a data warehouse. In this scenario the SEA might implement both the SREPublisher and SREConsumer services (in addition to LEAs implementing these services as well).

The *SREBrokerage* service allows for brokered exchanges between trading partners that do not know of one another. Its GetStudentRecordExchange method is called to request that records be retrieved for a given student and sending agency. Depending on the business rules and capabilities of the overall student record exchange system, a transaction with a brokerage can move data from a sending agency to a receiving agency either at the request of the sender or at the request of the receiver. Initiating a request from the receiving agency is by far the most common scenario.

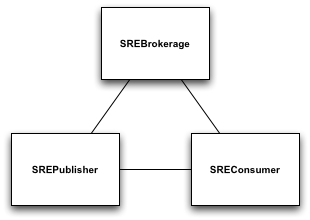


Figure . SRE Transaction Services

### Summary of Service Methods

The following tables summarize the Transaction Services and their methods. No service events are reported by any of the services in this document.

**SREPublisher**

|  |  |
| --- | --- |
| **Method** | **Description** |
| PublishStudentRecordExchange | Publish a StudentRecordExchange object set.  **Parameters:** The StateProvinceId of a student, a sending agency (the source of the student data), and a receiving agency (the destination for the student data).  **Result:** A composite object that contains a StudentRecordExchange and the objects it references: StudentDemographicRecord, StudentAcademicRecord, StudentSpecialEducationRecord, and StudentRecordPackage. |

**SREConsumer**

|  |  |
| --- | --- |
| **Method** | **Description** |
| ConsumeStudentRecordExchange | Consume a StudentRecordExchange object set for a student from a sending agency.  **Parameters:** The StateProvinceId of a student, sending agency (the source of the student data), and a receiving agency (the destination for the data); and a composite object that contains a StudentRecordExchange and the objects it references: StudentDemographicRecord, StudentAcademicRecord, StudentSpecialEducationRecord, and StudentRecordPackage.  **Result:** None |

**SREBrokerage**

|  |  |
| --- | --- |
| **Method** | **Description** |
| GetStudentRecordExchange | Initiates a brokered student record exchange between a sending agency and a receiving agency. The receiving agency is always the caller of this method.  **Parameters:** The StateProvinceId of a student, sending agency (the source of the student data), and receiving agency (the destination for the student data).  **Result:** A composite object that contains a StudentRecordExchange and the objects it references: StudentDemographicRecord, StudentAcademicRecord, StudentSpecialEducationRecord, and StudentRecordPackage. |

### Requesting Student Records Directly from a Trading Partner

When a trading partner wishes to request a StudentRecordExchange object set from another trading partner, it calls the PublishStudentRecordExchange method of the SREPublisher service. The method parameters include:

1. StateProvinceId of the student
2. StateProvinceId of the sending agency (the agency supplying the data)
3. StateProvinceId of the receiving agency (the agency requesting the data)

The SREPublisher service returns a SIF\_ServiceOutput message that contains a StudentRecordExchange object set, or in the case of error, a SIF\_Error element. The entire set of student records is returned as the single response to this method invocation. Because StudentRecordExchanges can be very large in size, the results may span multiple packets. The packetizing rules described by SIF Zone Services apply.

Figure 3 illustrates a requestor calling the PublishStudentRecordExchange method of a publisher agent that implements the SREPublisher service (step ➊). The publisher agent issues SIF\_Requests to its local zone to construct a StudentRecordExchange object set (step ➋) or otherwise obtains the data according to its business rules. The results are returned back to the requesting agent (step ➌).

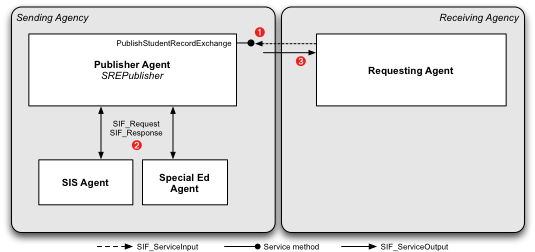


Figure . Student Record Exchange between Trading Partners

### Requesting Student Records via a Brokerage

When a trading partner wishes to request a StudentRecordExchange object set from a brokerage, it calls the GetStudentRecordExchange method of the SREBrokerage service. The method parameters include:

1. StateProvinceId of the student
2. StateProvinceId of the sending agency (the agency supplying the data)
3. StateProvinceId of the receiving agency (the agency requesting the data)

The brokerage works with the SREPublisher service of the trading partner representing the sending agency in order to obtain the requested data. The trading partner that’s contacted by the brokerage could be the LEA identified as the sending agency, or it could be a third party such as a data warehouse hosted by a SEA or other service provider. It is also possible that the brokerage itself can supply the student’s data from a local repository.

When the student’s records are available for delivery back to the requestor, or if the transaction has failed with an error, the brokerage responds with a SIF\_ServiceOutput message that contains either a set of StudentRecordExchange objects or a SIF\_Error element with error information. The entire set of student records is returned as the single response to this method invocation. Because StudentRecordExchanges can be very large in size, the results may span multiple packets. The packetizing rules described by SIF Zone Services apply.

If the brokerage cannot satisfy the request for any reason, it responds with a SIF\_ServiceOutput message with a SIF\_Error element. Possible reasons for failure include: the brokerage does not know how to get in touch with a trading partner representing the sending agency; that trading partner does not implement the SREPublisher service; a communication error occurred while calling the SREPublisher service of the trading partner; the service input parameters are invalid; and so on.

Figure 4 illustrates a requestor calling the GetStudentRecordExchange service method of a brokerage agent, which implements the SREBrokerage service (step ➊). The brokerage then calls the PublishStudentRecordExchange service method of a publisher agent that implements the SREPublisher service to obtain student records (step ➋). The publisher agent issues SIF\_Requests to its local zone to construct a StudentRecordExchange object set (step ➌) or otherwise obtains the data according to its business rules. The results are returned back to the brokerage agent (step ➍), and then back to the requestor (step ➎).

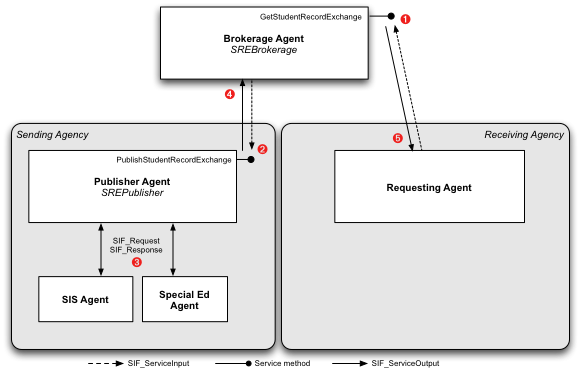


Figure . Requesting Student Records via Brokerage

### End-to-End Student Record Exchange from a Brokerage

A common feature of commercial brokerage solutions is to provide a central user interface from which registrars manage electronic transcript and student record exchanges. Such brokerages can act as the initiator of end-to-end student record exchanges, and therefore require a way to deliver student records to a receiving agency without the receiving agency having previously requested of those records. The SREConsumer service enables this interaction.

When a brokerage wishes to perform an end-to-end student record exchange, it first interacts with the SREPublisher service of the trading partner representing the sending agency in order to obtain the student’s records. The trading partner that’s contacted by the brokerage could be the LEA identified as the sending agency, or it could be a third party such as a data warehouse hosted by a SEA or other service provider. It is also possible that the brokerage itself can supply the student’s data from a local repository.

When the student’s records are available for delivery, the brokerage calls the ConsumeStudentRecordExchange method of the SREConsumer service of the trading partner that represents the receiving agency. If the receiving agency does not implement the SREConsumer service, it does not accept unrequested student record exchanges.

The method parameters include:

1. StateProvinceId of the student
2. StateProvinceId of the sending agency (the agency supplying the data)
3. StateProvinceId of the receiving agency (the agency receiving the data)
4. The StudentRecordExchange object set to deliver

The entire set of student records is provided as a parameter to the method. Because StudentRecordExchanges can be very large in size, the method invocation may span multiple SIF\_ServiceInput packets. The packetizing rules described by SIF Zone Services apply.

Figure 5 illustrates a brokerage calling the PublishStudentRecordExchange method of a publisher agent at the sending agency, which implements the SREPublisher service (step ➊). The publisher agent issues SIF\_Requests to its local zone to construct a StudentRecordExchange object set (step ➋) or otherwise obtains the data according to its business rules. The results are returned back to the brokerage agent (step ➌). The brokerage then calls the ConsumeStudentRecordExchange method of a consumer agent at the receiving agency (step ➍) to deliver the data to its destination. No data is returned in the response to this method (step ➎), but it may contain a SIF\_Error if an error occurred at the receiving agency.

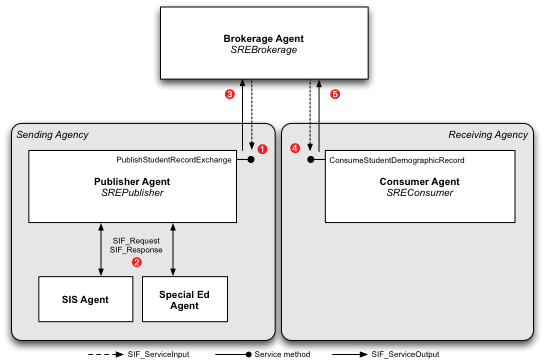


Figure . Broker-Initiated Student Record Exchange

## Consumer Services

Whereas the Transaction Services above describe how to move student data between trading partners and brokerages, the optional Consumer Services define how to consume or import student record exchange data into target systems such as student information systems. Agents that implement the Consumer Services unambiguously declare that they can consume StudentRecordExchange data in whole or part.

The StudentRecordExchange object is designed as a content package. It is comprised of several types of records (e.g. StudentDemographicRecord, StudentAcademicRecord, etc.), each of which may be provided by a different authoritative source in a zone. When consuming StudentRecordExchange data into a zone, then, it is possible that different systems will be interested in different parts of that content package. An SIS may need to import the StudentDemographicRecord into its database, and a separate special education package import StudentSpecialEducationRecord.

Consumer Services are comprised of:

* SREStudentDemographicRecordConsumer
* SREStudentAcademicRecordConsumer
* SRESpecialEducationRecordConsumer

All SIF Agents that supply student demographic, academic, or special education records are encouraged to implement the appropriate Consumer Services. Doing so enables the agent to participate in state-wide student record exchange initiatives, particularly LEA-to-LEA student transfer. In addition, these agents are encouraged to provide SIF\_Request support for the applicable StudentDemograpicRecord, StudentAcadmicRecord, or StudentSpecialEducationRecord objects that apply to them. Doing so will make it considerably easier for intermediary publisher agents (i.e. those that implement the SREPublisher service) to obtain the data needed to satisfy a request.

If an application is capable of consuming an entire StudentRecordExchange object, it can do so via the SREConsumer service and choose to not implement the individual record-specific Consumer Services. However, as most applications are only interested or capable of processing a subset of StudentRecordExchange, each should implement the specific Consumer Services that it supports as a best practice. This makes it clear the role each SIF Agent in a zone plays in consuming StudentRecordExchange data. It should also be noted that the SREConsumer service is part of Transaction Services, which are intended to be implemented by intermediary agents in a comprehensive student record exchange solution. It is therefore envisioned that intermediary agents will implement the SREConsumer service, delegating to the various Consumer Services in a zone to actually import data into target systems.

### Rules for Calling Consumer Services

When an intermediary agent that implements the SREConsumer service receives a StudentRecordExchange object set and wishes to import the data into target systems, it follows the rules below.

By employing these rules, it is possible for system integrators to install consumer agents of various types in a local district zone, such that each agent can import all or part of a StudentRecordExchange object set received as part of a larger student record exchange transaction. Agents that wish to import all of a StudentRecordExchange object set can implement the single SREConsumer service instead of the individual record-level Consumer Services. Agents that wish to import StudentRecordPackage data, for which no Consumer Service exists, should also implement the SREConsumer service.

For each local zone the intermediary agent is connected to:

1. It obtains a SIF\_ZoneStatus object to determine the agents in the zone that implement the SREConsumer service or the individual record-level Consumer Services.
2. For each agent that implements the SREConsumer service (excluding itself), the intermediary agent invokes the ConsumeStudentRecordExchange method.
3. For each agent that implements the SREStudentDemographicRecordConsumer service, the intermediary agent invokes the ConsumeStudentDemographicRecord method, supplying only the StudentDemographicRecord portion of the student data.
4. For each agent that implements the SREStudentAcademicRecordConsumer service, the intermediary agent invokes the ConsumeStudentAcademicRecord method, supplying only the StudentAcademicRecord portion of the student data.
5. For each agent that implements the SREStudentSpecialEducationRecordConsumer service, the intermediary agent invokes the ConsumeStudentSpecialEducationRecord method, supplying only the StudentSpecialEducationRecord portion of the student data.

### Consuming Student Records into Target Applications

Figure 7 illustrates how the Consumer Services can be employed in a comprehensive student record exchange solution to import data received from an intermediary agent into target applications like the student information system. Here, the LEA-to-LEA Student Records Transfer use case is demonstrated.

A consumer agent at the receiving agency calls the GetStudentRecordExchange method of the brokerage agent’s SREBrokerage service (step ➊). Student records are obtained from a publisher agent at the sending agency via the SREPublisher service (step ➋). The publisher agent issues SIF\_Requests to its local zone to construct a StudentRecordExchange object set (step ➌) and returns the data back to the requestor (step ➍ and ➎). The consumer agent then consults the SIF\_ZoneStatus object of its local district zone and learns that two applications are capable of consuming student record exchange data. It asks the student information system’s agent, which implements the SREStudentDemographicRecordConsumer and SREStudentAcademicRecordConsumer services, to import those portions of the student data (step ➏ and ➐). Next, it asks the special education package, which implements the SREStudentSpecialEducationRecordConsumer service, to import the special education record (step ➑).

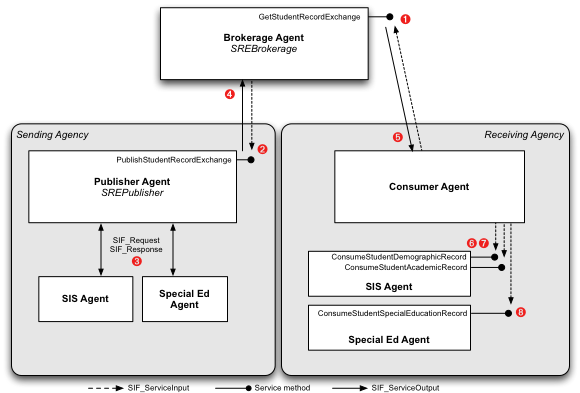


Figure . Consumer Services used in LEA-to-LEA Student Transfer

Figure 8 depicts a variation of the above, where the SIS Agent implements the SREConsumer service instead of the individual record-level Consumer Services. It might do this, for example, if it needs to process StudentRecordPackage data, for which no record-level Consumer Service exists.

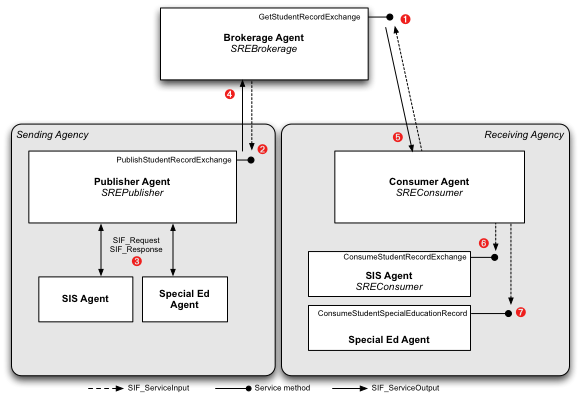


Figure . Consumer Services and SREConsumer used in LEA-to-LEA Student Transfer

## Web Services

This document presents a set of conceptual services for student record exchange, with a specific binding of those services to the SIF Zone Services infrastructure. SIF Zone Services satisfy the needs of trading partners that have a local SIF infrastructure – that is, local zones and agents that can communicate with the zones and agents of other trading partners or brokerages. However, many Student Record Exchange uses cases involve trading partners that are outside of the pK-12 education space, and as such these organizations may not have a SIF infrastructure. In a LEA-to-Postsecondary exchange, for example, the publishing of student records may occur between the LEA and brokerage using SIF Zone Services, while the delivery of student records to the Postsecondary institution might occur over some other web service protocol like SOAP.

The following use cases from the Student Record Exchange Task Force’s Business Case fall into this category:

* High School to Postsecondary
* Postsecondary Institution to High School
* High School to Employer
* High School to Postsecondary Lending Organization
* High School to Scholarship Organization

And possibly these as well, although a SIF-based infrastructure might exist for some:

* School/District to Migrant Education Program
* Virtual School to School/District
* School/District to State
* High School to NCAA Clearinghouse

Currently, there is no Web Services binding for the services defined in this document. The Student Record Exchange Task Force anticipates adding such a binding so as soon as SIFA has more clearly defined a direction for Web Services. Presumably, the approach taken will resemble the existing Vertical Reporting Web Service specification.

# Use Cases

## Use Case SRE01 – Brokered LEA-to-LEA Student Records Transfer

## Use Case SRE02 – Brokered LEA-to-Postsecondary Transcript Exchange

# Service Definition

## The StudentRecordExchangeData Composite Object

All of the service methods defined here either return or accept as input a StudentRecordExchange *object set*, which is comprised of a single StudentRecordExchange object and all of the dependent objects it references in its Records and OtherRecords elements. Because a StudentRecordExchange object set can be potentially very large, it is necessary to packetize these objects across multiple SIF\_ServiceInput or SIF\_ServiceOutput messages. The composite StudentRecordExchangeData object is designed for this purpose.

StudentRecordExchangeData is always declared to be a Repeatable element in service messages. An instance of this element should be created for each StudentRecordExchange, StudentDemographicRecord, StudentAcademicRecord, StudentSpecialEducationRecord, or StudentRecordPackage object in the set. Each StudentRecordExchangeData instance must only contain one of these elements.

| **Element** | **Char** | **Description** |
| --- | --- | --- |
| StudentRecordExchangeData | M |  |
| StudentRecordExchangeData/StudentRecordExchange | C | The StudentRecordExchange object. At most one of this or one of the other Conditional elements must be provided in a StudentRecordExchangeData instance. |
| StudentRecordExchangeData/StudentDemographicRecord | C | The StudentDemographicRecord object referenced by the StudentRecordExchange/Records/StudentDemographicRecordRefId element. At most one of this or one of the other Conditional elements must be provided in a StudentRecordExchangeData instance. |
| StudentRecordExchangeData/StudentAcademicRecord | C | The StudentAcademicRecord object referenced by the StudentRecordExchange/Records/StudentAcademicRecordRefId element. At most one of this or one of the other Conditional elements must be provided in a StudentRecordExchangeData instance. |
| StudentRecordExchangeData/StudentSpecialEducationRecord | C | The StudentSpecialEducationRecord object referenced by the StudentRecordExchange/Records/StudentSpecialEducationRecordRefId element. At most one of this or one of the other Conditional elements must be provided in a StudentRecordExchangeData instance. |
| StudentRecordExchangeData/StudentRecordPackage | C | A StudentRecordPackage object referenced by the StudentRecordExchange/Records/OtherRecords element. At most one of this or one of the other Conditional elements must be provided in a StudentRecordExchangeData instance. |

Example XML

<StudentRecordExchangeData>

<StudentRecordExchange RefId="0D015F74DAB645FD92EFA8F43F2D79C3">  
 <StateProvinceId>10009600</StateProvinceId>  
 <Records>  
 <StudentDemographicRecordRefId>A15484ED564995254A4568EFFC5100BD</StudentDemographicRecordRefId>  
 <StudentAcademicRecordRefId>BB181B05598C46D2B8D533483D91392E</StudentAcademicRecordRefId>  
 <StudentSpecialEducationRecordRefId>AA81B05598C46D2B8D533483D9139EEF</StudentSpecialEducationRecordRefId>  
 <OtherRecords>  
 <StudentRecordPackageRefId>EE5484ED564995254A4568EFFC510044BD</StudentRecordPackageRefId>  
 </OtherRecords>  
 </Records>  
 </StudentRecordExchange>  
 </StudentRecordExchangeData>  
 <StudentRecordExchangeData>

<StudentDemographicRecord   
 RefId="A15484ED564995254A4568EFFC5100BD”  
 SIF\_RefObject="StudentRecordExchange”  
 SIF\_RefId="0D015F74DAB645FD92EFA8F43F2D79C3">  
 <ReportingDate>2008-11-07</ReportingDate>  
 <StudentPersonalData>  
 <LocalId>88495</LocalId>  
 <StateProvinceId>10009600</StateProvinceId>  
 <Name Type="06”>  
 <LastName>Petersen</LastName>  
 <FirstName>Eric</FirstName>  
 </Name>  
 </StudentPersonalData>  
 </StudentDemographicData>  
 </StudentRecordExchangeData>

<StudentRecordExchangeData>

<StudentAcademicRecord   
 RefId="BB181B05598C46D2B8D533483D91392E”  
 SIF\_RefObject="StudentRecordExchange”  
 SIF\_RefId="0D015F74DAB645FD92EFA8F43F2D79C3">  
 ...  
 </StudentAcademicRecord>  
 </StudentRecordExchangeData>

<StudentRecordExchangeData>

<StudentSpecialEducationRecord   
 RefId="AA81B05598C46D2B8D533483D9139EEF”  
 SIF\_RefObject="StudentRecordExchange”  
 SIF\_RefId="0D015F74DAB645FD92EFA8F43F2D79C3">  
 ...  
 </StudentSpecialEducationRecord>  
 </StudentRecordExchangeData>

<StudentRecordExchangeData>

<StudentRecordPackage  
 RefId="EE5484ED564995254A4568EFFC510044BD”  
 SIF\_RefObject="StudentRecordExchange”  
 SIF\_RefId="0D015F74DAB645FD92EFA8F43F2D79C3">  
 ...  
 </StudentRecordPackage>  
 </StudentRecordExchangeData>

### SIF\_MaxBufferSize and StudentRecordExchangeData

The minimum SIF\_MaxBufferSize recommended for SIF Zone Services is 64K. In order for service messages to packetize StudentRecordExchangeData elements, the buffer size used must be large enough to accommodate the largest element of that object. StudentRecordPackage objects in particular can be larger than 64K if they contain Base64-encoded binary files such as photographs and documents that are transmitted with a student’s transcript or records.

### ExtendedParameters

Each service method defined here accepts a minimum of three input parameters: StudentId, SendingAgencyId, and ReceivingAgencyId. While these parameters are sufficient for all implementations to identify the student and agencies involved in a student record exchange, systems may need to be able to convey additional vendor-specific parameters. The optional ExtendedParameters element can be used for this purpose.

| **Element** | **Char** | **Description** |
| --- | --- | --- |
| ExtendedParameters | M | A list of ExtendedParameter elements |
| ExtendedParameters/ExtendedParameter | OR | An arbitrary parameter with a string name and value |
| ExtendedParameters/ExtendedParameter/SIF\_Name | M | The name of the parameter |
| ExtendedParameters/ExtendedParameter/SIF\_Value | M | The value of the parameter |

Example XML

<ExtendedParameters>  
 <ExtendedParameter>  
 <SIF\_Name>Options</SIF\_Name>  
 <SIF\_Value>0x0001</SIF\_Value>  
 </ExtendedParameter>  
 <ExtendedParameter>  
 <SIF\_Name>ExchangeType</SIF\_Name>  
 <SIF\_Value>Transcript</SIF\_Value>  
 </ExtendedParameter>  
</ExtendedParameters>

## SREBrokerage Service

### Service Methods

| **Request Message** | **Response Message** | **Description** |
| --- | --- | --- |
| GetStudentRecordExchange | GetStudentRecordExchangeResponse | Requests from the brokerage a StudentRecordExchange object set for a given student and sending agency. |

### Service Events

The SREBrokerage Service defines no service events.

### Detailed Message Definition

The GetStudentRecordExchange message requests from the brokerage a StudentRecordExchange object set for a given student and sending agency.

### GetStudentRecordExchange Message

| **Element** | **Char** | **Description** |
| --- | --- | --- |
| GetStudentRecordExchange | M |  |
| StudentId | M | The StateProvinceId of the student |
| SendingAgencyId | M | The StateProvinceId of the sending agency |
| ReceivingAgencyId | M | The StateProvinceId of the receiving agency |
| ExtendedParameters | O | Optional list of additional implementation-dependent parameters |

Example XML

<GetStudentRecordExchange>  
 <StudentId>10009600</StudentId>

<SendingAgencyId>884</SendingAgencyId>

<ReceivingAgencyId>601</ReceivingAgencyId>  
 <ExtendedParameters>  
 <ExtendedParameter>  
 <SIF\_Name>Options</SIF\_Name>  
 <SIF\_Value>0x0001</SIF\_Value>  
 </ExtendedParameter>  
 <ExtendedParameter>  
 <SIF\_Name>ExchangeType</SIF\_Name>  
 <SIF\_Value>Transcript</SIF\_Value>  
 </ExtendedParameter>  
 </ExtendedParameters>

</GetStudentRecordExchange>

### GetStudentRecordExchangeResponse Message

| **Element** | **Char** | **Description** |
| --- | --- | --- |
| GetStudentRecordExchangeResponse | M |  |
| StudentRecordExchangeData | MR | The StudentRecordExchange object set for the student and sending agency that was requested by the GetStudentRecordExchange method. The object set is contained in a composite StudentRecordExchangeData object. To allow for packetizing of StudentRecordExchange object sets, there should be one instance of the StudentRecordExchangeData object for each object it contains. |

Example XML

<GetStudentRecordExchangeResponse>

<StudentRecordExchangeData>

<StudentRecordExchange RefId="0D015F74DAB645FD92EFA8F43F2D79C3">  
 <StateProvinceId>10009600</StateProvinceId>  
 <Records>  
 <StudentDemographicRecordRefId>A15484ED564995254A4568EFFC5100BD</StudentDemographicRecordRefId>  
 <StudentAcademicRecordRefId>BB181B05598C46D2B8D533483D91392E</StudentAcademicRecordRefId>  
 </Records>  
 </StudentRecordExchange>  
 </StudentRecordExchangeData>  
 <StudentRecordExchangeData>

<StudentDemographicRecord   
 RefId="A15484ED564995254A4568EFFC5100BD”  
 SIF\_RefObject="StudentRecordExchange”  
 SIF\_RefId="0D015F74DAB645FD92EFA8F43F2D79C3">  
 <ReportingDate>2008-11-07</ReportingDate>  
 <StudentPersonalData>  
 <LocalId>88495</LocalId>  
 <StateProvinceId>10009600</StateProvinceId>  
 <Name Type="06”>  
 <LastName>Petersen</LastName>  
 <FirstName>Eric</FirstName>  
 </Name>  
 </StudentPersonalData>  
 </StudentDemographicData>  
 </StudentRecordExchangeData>

<StudentRecordExchangeData>

<StudentAcademicRecord   
 RefId="BB181B05598C46D2B8D533483D91392E”  
 SIF\_RefObject="StudentRecordExchange”  
 SIF\_RefId="0D015F74DAB645FD92EFA8F43F2D79C3">  
 ...  
 </StudentAcademicRecord>  
 </StudentRecordExchangeData>

</GetStudentRecordExchangeResponse>

## SREPublisher Service Definition

### Service Methods

| **Request Message** | **Response Message** | **Description** |
| --- | --- | --- |
| PublishStudentRecordExchange | PublishStudentRecordExchangeResponse | Requests that a StudentRecordExchange object set be published for a given student and sending agency |

### Service Events

The SREPublisher Service defines no service events.

### Detailed Message Definition

The PublishStudentRecordExchange message requests that a StudentRecordExchange object set be published for a given student and sending agency.

### PublishStudentRecordExchange Message

| **Element** | **Char** | **Description** |
| --- | --- | --- |
| PublishStudentRecordExchange | M |  |
| StudentId | M | The StateProvinceId of the student |
| SendingAgencyId | M | The StateProvinceId of the sending agency |
| ReceivingAgencyId | M | The StateProvinceId of the receiving agency |
| ExtendedParameters | O | Optional list of additional implementation-dependent parameters. |

Example XML

<PublishStudentRecordExchange>  
 <StudentId>10009600</StudentId>

<SendingAgencyId>884</SendingAgencyId>

<ReceivingAgencyId>601</ReceivingAgencyId>  
 <ExtendedParameters>  
 <ExtendedParameter>  
 <SIF\_Name>Options</SIF\_Name>  
 <SIF\_Value>0x0001</SIF\_Value>  
 </ExtendedParameter>  
 <ExtendedParameter>  
 <SIF\_Name>ExchangeType</SIF\_Name>  
 <SIF\_Value>Transcript</SIF\_Value>  
 </ExtendedParameter>  
 </ExtendedParameters>

</PublishStudentRecordExchange>

### PublishStudentRecordExchangeResponse Message

| **Element** | **Char** | **Description** |
| --- | --- | --- |
| PublishStudentRecordExchangeResponse | M |  |
| StudentRecordExchangeData | MR | The StudentRecordExchange object set for the student and sending agency that was requested by the PublishStudentRecordExchange method. The object set is contained in a composite StudentRecordExchangeData object. To allow for packetizing of StudentRecordExchange object sets, there should be one instance of the StudentRecordExchangeData object for each object it contains. |

Example XML

<PublishStudentRecordExchangeResponse>

<StudentRecordExchangeData>

<StudentRecordExchange RefId="0D015F74DAB645FD92EFA8F43F2D79C3">  
 <StateProvinceId>10009600</StateProvinceId>  
 <Records>  
 <StudentDemographicRecordRefId>A15484ED564995254A4568EFFC5100BD</StudentDemographicRecordRefId>  
 <StudentAcademicRecordRefId>BB181B05598C46D2B8D533483D91392E</StudentAcademicRecordRefId>  
 </Records>  
 </StudentRecordExchange>  
 </StudentRecordExchangeData>  
 <StudentRecordExchangeData>

<StudentDemographicRecord   
 RefId="A15484ED564995254A4568EFFC5100BD”  
 SIF\_RefObject="StudentRecordExchange”  
 SIF\_RefId="0D015F74DAB645FD92EFA8F43F2D79C3">  
 <ReportingDate>2008-11-07</ReportingDate>  
 <StudentPersonalData>  
 <LocalId>88495</LocalId>  
 <StateProvinceId>10009600</StateProvinceId>  
 <Name Type="06”>  
 <LastName>Petersen</LastName>  
 <FirstName>Eric</FirstName>  
 </Name>  
 </StudentPersonalData>  
 </StudentDemographicData>  
 </StudentRecordExchangeData>

<StudentRecordExchangeData>

<StudentAcademicRecord   
 RefId="BB181B05598C46D2B8D533483D91392E”  
 SIF\_RefObject="StudentRecordExchange”  
 SIF\_RefId="0D015F74DAB645FD92EFA8F43F2D79C3">  
 ...  
 </StudentAcademicRecord>  
 </StudentRecordExchangeData>

</PublishStudentRecordExchangeResponse>

## SREConsumer Service Definition

### Service Methods

| **Request Message** | **Response Message** | **Description** |
| --- | --- | --- |
| ConsumeStudentRecordExchange | ConsumeStudentRecordExchangeResponse | Requests that a StudentRecordExchange object set be consumed by a receiving agency. For example, if a brokerage initiates an exchange between a sending agency and a receiving agency, it uses this service to deliver the results to the receiving agency. |

### Service Events

The SREConsumer Service defines no service events.

### Detailed Message Definition

The ConsumeStudentRecordExchange message requests that a StudentRecordExchange object for a given student and sending agency be consumed.

### ConsumeStudentRecordExchange Message

| **Element** | **Char** | **Description** |
| --- | --- | --- |
| ConsumeStudentRecordExchange | M |  |
| StudentId | M | The StateProvinceId of the student. In the case where a SIF\_ServiceInput message spans multiple packets, each packet must provide this element. |
| SendingAgencyId | M | The StateProvinceId of the sending agency. In the case where a SIF\_ServiceInput message spans multiple packets, each packet must provide this element. |
| ReceivingAgencyId | M | The StateProvinceId of the receiving agency. In the case where a SIF\_ServiceInput message spans multiple packets, each packet must provide this element. |
| ExtendedParameters | O | Optional list of additional implementation-dependent parameters. |
| StudentRecordExchangeData | MR | The StudentRecordExchange object set for a single student and sending agency. The object set is contained in a composite StudentRecordExchangeData object. To allow for packetizing of StudentRecordExchange object sets, there should be one instance of the StudentRecordExchangeData object for each object it contains. |

Example XML

<ConsumeStudentRecordExchange>  
 <StudentId>10009600</StudentId>

<SendingAgencyId>884</SendingAgencyId>

<ReceivingAgencyId>601</ReceivingAgencyId>

<ExtendedParameters>  
 <ExtendedParameter>  
 <SIF\_Name>Options</SIF\_Name>  
 <SIF\_Value>0x0001</SIF\_Value>  
 </ExtendedParameter>  
 <ExtendedParameter>  
 <SIF\_Name>ExchangeType</SIF\_Name>  
 <SIF\_Value>Transcript</SIF\_Value>  
 </ExtendedParameter>  
 </ExtendedParameters>

<StudentRecordExchangeData>

<StudentRecordExchange RefId="0D015F74DAB645FD92EFA8F43F2D79C3">  
 <StateProvinceId>10009600</StateProvinceId>  
 <Records>  
 <StudentDemographicRecordRefId>A15484ED564995254A4568EFFC5100BD</StudentDemographicRecordRefId>  
 <StudentAcademicRecordRefId>BB181B05598C46D2B8D533483D91392E</StudentAcademicRecordRefId>  
 </Records>  
 </StudentRecordExchange>  
 </StudentRecordExchangeData>  
 <StudentRecordExchangeData>

<StudentDemographicRecord   
 RefId="A15484ED564995254A4568EFFC5100BD”  
 SIF\_RefObject="StudentRecordExchange”  
 SIF\_RefId="0D015F74DAB645FD92EFA8F43F2D79C3">  
 <ReportingDate>2008-11-07</ReportingDate>  
 <StudentPersonalData>  
 <LocalId>88495</LocalId>  
 <StateProvinceId>10009600</StateProvinceId>  
 <Name Type="06”>  
 <LastName>Petersen</LastName>  
 <FirstName>Eric</FirstName>  
 </Name>  
 </StudentPersonalData>  
 </StudentDemographicData>  
 </StudentRecordExchangeData>

<StudentRecordExchangeData>

<StudentAcademicRecord   
 RefId="BB181B05598C46D2B8D533483D91392E”  
 SIF\_RefObject="StudentRecordExchange”  
 SIF\_RefId="0D015F74DAB645FD92EFA8F43F2D79C3">  
 ...  
 </StudentAcademicRecord>  
 </StudentRecordExchangeData>

</ConsumeStudentRecordExchange>

### ConsumeStudentRecordExchangeResponse Message

The ConsumeStudentRecordExchange service method does not return any data as a result. However, to allow for error information to be relayed back to the caller of the method, an object is defined. If the ConsumeStudentRecordExchange method is successful, the SIF\_ServiceOutput message will contain an empty object. If an error occurred, the SIF\_ServiceOutput message will contain a SIF\_Error element that describes the error condition.

| **Element** | **Char** | **Description** |
| --- | --- | --- |
| ConsumeStudentRecordExchangeResponse | M | This object is empty. |

Example XML

<ConsumeStudentRecordExchangeResponse/>

## SRERecordTypeConsumer Service Pattern

The SRE*RecordType*Consumer Service pattern is used by each of the record-level Consumer Services, where *RecordType* is the name of a record contained in a StudentRecordExchange object. Note StudentRecordPackage is excluded from this pattern.

### Service Methods

| **Request Message** | **Response Message** | **Description** |
| --- | --- | --- |
| Consume*RecordType* | Consume*RecordType*Response | Requests that a *RecordType* object for a given student be consumed. |

### Service Events

The SRE*RecordType*Consumer Service Pattern defines no service events.

### Detailed Message Definition

The Consume*RecordType* message requests that a *RecordType* object for a given student be consumed.

### Consume*RecordType* Message

| **Element** | **Char** | **Description** |
| --- | --- | --- |
| Consume*RecordType* | M |  |
| StudentId | M | The StateProvinceId of the student. In the case where a SIF\_ServiceInput message spans multiple packets, each packet must provide this element. |
| SendingAgencyId | M | The StateProvinceId of the sending agency. In the case where a SIF\_ServiceInput message spans multiple packets, each packet must provide this element. |
| ReceivingAgencyId | M | The StateProvinceId of the receiving agency. In the case where a SIF\_ServiceInput message spans multiple packets, each packet must provide this element. |
| ExtendedParameters | O | Optional list of additional implementation-dependent parameters. |
| *RecordType* | M | The *RecordType* object |

Example XML

<ConsumeStudentDemographicRecord>  
 <StudentId>10009600</StudentId>

<SendingAgencyId>884</SendingAgencyId>

<ReceivingAgencyId>601</ReceivingAgencyId>  
 <ExtendedParameters>  
 <ExtendedParameter>  
 <SIF\_Name>Options</SIF\_Name>  
 <SIF\_Value>0x0001</SIF\_Value>  
 </ExtendedParameter>  
 <ExtendedParameter>  
 <SIF\_Name>ExchangeType</SIF\_Name>  
 <SIF\_Value>Transcript</SIF\_Value>  
 </ExtendedParameter>  
 </ExtendedParameters>

<StudentDemographicRecord   
 RefId="A15484ED564995254A4568EFFC5100BD”  
 SIF\_RefObject="StudentRecordExchange”  
 SIF\_RefId="0D015F74DAB645FD92EFA8F43F2D79C3">  
 <ReportingDate>2008-11-07</ReportingDate>  
 <StudentPersonalData>  
 <LocalId>88495</LocalId>  
 <StateProvinceId>10009600</StateProvinceId>  
 <Name Type="06”>  
 <LastName>Petersen</LastName>  
 <FirstName>Eric</FirstName>  
 </Name>  
 </StudentPersonalData>  
 </StudentDemographicRecord>  
</ConsumeStudentDemographicData>

### Consume*RecordType*Response Message

The Consume*RecordType* service method does not return any data as a result. However, to allow for error information to be relayed back to the caller of the method, an object is defined. If the Consume*RecordType*Exchange method is successful, the SIF\_ServiceOutput message will contain an empty object. If an error occurred, the SIF\_ServiceOutput message will contain a SIF\_Error element that describes the error condition.

| **Element** | **Char** | **Description** |
| --- | --- | --- |
| Consume*RecordType*Response | M | This object is empty. |
|  |  |  |

Example XML

<ConsumeStudentDemographicRecordResponse/>